UNLIMITED DISTRIBUTION

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

CRAD SIGNAL AND INFORMATION PROCESSING WORKING GROUP WORKSHOP ON NEURAL NETWORKS

AT

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC

6 - 7 MAY 1992

VOLUME 1

UNCLASSIFIED PROCEEDINGS

(Volume 2 - Classified proceedings - available under separate cover)

B.A. McArthur (editor)

February 1993

Approved by A.T. Ashley
Head / Signal Processing Section

Distribution Approved by C.W. Bright

C.W. Bright
Director / Sonar Division

TECHNICAL COMMUNICATION 93/305

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

P129006.PDF [Page: 4 of 4]

106-04130/

ABSTRACT

The Signal and Information Processing Working Group (SIPWG) Workshop on Neural Networks, sponsored by the Chief of Research and Development (CRAD) Branch of the Department of National Defence (DND), was held on 6-7 May 1992 at the Defence Research Establishment Atlantic. The objective of the workshop was to provide the Canadian defence research establishments and military colleges with a forum for the exchange of information related to the theory, design, and application of neural networks. The workshop format consisted of presentations, organized into four half-day sessions, and group discussions, one following each set of presentations. Thirteen presentations were given at the workshop. Nine papers described defence-related applications of neural networks, including radar, sonar, speech, and sensor fusion, and the four remaining papers were devoted to neural network theory and design.

RÉSUMÉ

L'atelier sur les réseaux neuronaux du groupe de travail sur le traitement des signaux et de l'information (SIPWG), commandité par le Chef – Recherche et développement (CR DÉV) du ministère de la Défense nationale, a eu lieu du 6 au 7 mai 1992, au Centre de recherches pour la défense Atlantique. Cet atelier a été conçu comme un forum pour l'échange d'information sur la théorie, la conception et l'application des réseaux neuronaux, destiné aux collèges militaires et aux centres de recherches pour la défense du Canada. L'atelier était formé d'exposés, répartis en quatre séances d'une demi-journée, chaque exposé étant suivi d'une discussion. Treize communications ont été présentées pendant l'atelier. Neufs communications traitaient de l'application des réseaux neuronaux dans le secteur de la défense, y compris le radar, le sonar, la parole et la fusion des donées des capteurs divers, les quatre autres communications portant sur la théorie et la conception des réseaux neuronaux.

THE APPLICATION OF A RADIAL BASIS FUNCTION NEURAL NETWORK FOR RADAR TRACKING OVER THE SEA

by

Henry Leung and Eloi Bossé

Surface Radar Section
Radar Division
Defence Research Establishment Ottawa
Ottawa, Ontario
Canada K1A 0Z4

Abstract

A low-angle tracking technique based on the use of a radial basis function neural network is presented. Computer simulations show that the new technique is capable of tracking both stationary and moving targets under low signal-to-noise ratio situations with a high accuracy. As well, the performance of the new technique is tested using real-life data. The results demonstrate the robustness and effectiveness of the new algorithm in terms of its independence of array errors and of the nature of the noise background.

1 Introduction

For decades, a considerable amount of research has been devoted to a near-classic problem which is commonly referred to as "low-angle radar tracking". It is primarily a multipath problem, wherein reflections of radar signals from the sea surface enter the main beam of a shipborne radar and seriously degrade its performance. When the height of a target is such that the angular separation between the direct and indirect paths, along which the radar signal travels from the target to the radar, is less than 0.8 antenna beamwidths (BWs), conventional monopulse techniques will produce large tracking errors.

Because of this limitation of monopulse radars, many signal processing techniques have been developed for tracking low-angle targets. Much of the research work to date has been concentrated in the area of array signal processing, wherein superresolution algorithms have been conceived, developed, tested and accepted or rejected based on their effectiveness and robustness in separating a target from its image and then providing an accurate estimate of the target's angle-of-arrival. Most, if not all, conventional superresolution techniques employed in low-angle radar tracking are based on linear modelling; i.e., the output of a beamformer is linearly related to the outputs of the array antenna. Linear models, in terms of statistics, only make use of the first and second order moment information (e.g. the mean and the variance) of the data. Higher order information is thrown away during the processing. In the case of low-angle tracking, however, the diffuse multipath is described to be coloured and K-distributed [1]. Consequently, these algorithms may not offer a satisfactory performance for low angle tracking over rough sea surfaces.

Accordingly, we introduce a technique which has been developed based on a radial basis function (RBF) neural network [2]. This technique, in conjunction with a comprehensive multipath model, can be used to achieve a resolution that is an order of magnitude greater than that achieved using conventional superresolution processing. In addition, it alleviates the ambiguity problem that other superresolution techniques encounter when a single frequency is used. These significant enhancements are brought about by replacing linear modelling with non-linear modelling.

Recently, neural networks have drawn a lot of attention from the signal processing community

[3,4]. In an attempt to reproduce the capability of a human brain, a neural network consists of a multilayer nonlinear network that is able to decode the higher order information contained in the input data. A neural network tries to reproduce human intellectual capabilities through a "learning" process in which the error between the input-output pair is minimized by varying the weights of the interconnections between the input and output layers. Here, the RBF neural network is applied to low-angle target tracking. The advantages of using an RBF network are twofold. First, it is capable of performing an accurate multivariate approximation even in an ill-posed environment. Second, the optimal weights in the RBF network can be estimated in a linear fashion, and hence is has a much faster convergent rate for learning than many other neural net structures such as the backpropagation multilayer network[4].

2 The Tracking Technique Based on RBF

The signal model to be used here is mainly based on the one described in [5]. This model makes use of geometrical information and a priori knowledge of a number of other physical parameters. These parameters include the refractivity gradient, the reflection coefficient, the specular and diffuse scattering coefficients, and the divergence factor. In order to be consistent with our experimental setup, a bistatic radar signal model is considered. The model is based on one-way transmission, which, despite its simplicity, is useful for testing the performance of spectral estimation techniques and is suitable for carrying out experimental investigations. It is much easier and less expensive to set up a beacon system and a narrow-band receiving array than to implement a radar system with its much more complicated transmission and reception system. The one-way transmission model is easily extended to a two-way model and the wide-band channel can in principle be treated as an ensemble of contiguous narrow-band channels.

The geometry of a simple flat earth model for multipath is illustrated in Fig. 1. By taking into account of the curvature of signal path due to refraction and the curvature of the Earth itself, the received signal at the k^{th} element of the array is given by

$$s_k = \gamma e^{j\psi} e^{j\kappa(r-(h'-z_k')^2/2r)}\left[1+\rho e^{j(\phi+2\kappa h' z_k'/r)}\right] \qquad (1)$$

where $\kappa = 2\pi / \lambda$ is the wavenumber

$$h' = h - r_2^2/2R_e$$

$$z_4' = z_4 - r_1^2/2R_e$$

$$R_e = R\,(1+6.37\times10^{-3}\,dN/dh)^{-1}$$

R is earth radius.

In (1), the term $\gamma e^{j\phi}$ is the complex amplitude of the radar signal and the term $\rho e^{j\phi}$ is the overall reflection coefficient. It generally consists of three factors, namely the reflection coefficient for a plane surface, Γ, the divergence factor due to curved surface, D, and the specular scattering coefficient S; i.e.

$$\rho e^{j\phi} = \Gamma \cdot D \cdot S \tag{2}$$

The approximate expressions for the Fresnel reflection coefficient for a smooth plane surface are given by

$$\Gamma_v = \frac{\sqrt{\epsilon_c}\sin\psi - 1}{\sqrt{\epsilon_c}\sin\psi + 1} \tag{3}$$

for vertical polarization and

$$\Gamma_h = \frac{\sin\psi - \sqrt{\epsilon_c}}{\sin\psi + \sqrt{\epsilon_c}} \tag{4}$$

for horizontal polarization, where ψ is the grazing angle and ϵ_c is the complex dielectric constant of the reflecting medium. The derivation of the grazing angle is based on the geometrical information. The divergence factor is approximated by

$$D = \left[1 + \frac{2r_1 r_2}{R_e r \psi}\right]^{-\frac{1}{2}} \tag{5}$$

The root-mean-squared specular scattering coefficient S is given by

$$S = e^{-\mu} \tag{6}$$

where

$$\mu = \begin{cases} 2[2\pi\eta]^2 & \eta \le 0.1 \text{ radian} \\ 0.165\eta^2 + 7.42\eta + 0.0468 & \text{otherwise} \end{cases} \tag{7}$$

and η is the surface roughness factor given by

$$\eta = \frac{\sigma_H \psi}{\lambda} \tag{8}$$

In (8), σ_H is the rms value of the sea wave height and λ is the wavelength of the microwave signal.

In practice, the received signal is contaminated with noise; i.e. the total received signal is

$$x_k = s_k + n_k \tag{9}$$

where n_k represents an arbitrary noise field.

In the RBF approach, the input/output relationship is modelled by a RBF neural network; i.e.

$$y(k) = \sum_{n=1}^{N} w_n \Phi_n(\|x - c_n\|) \tag{10}$$

where $\Phi_n(\|x-c_n\|)$ is a continuous function called the "basis function" and $\|\cdot\|$ denotes the norm in Euclidean space. The vectors c_n, n=1,..., N, are referred to as the centres of the basis functions. $\{w_n, n=1,...,N\}$ are the coefficients of the linear equation. For M snapshots of array inputs, a set of linear equations are formed based on (10)

$$\begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_M \end{bmatrix} = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} \begin{bmatrix} w_1 \\ \cdot \\ \cdot \\ \cdot \\ w_N \end{bmatrix} \tag{11}$$

where

$$A_{mn} = \Phi_{mn}(\|x_m - c_n\|) \tag{12}$$

and x_m is the m^{th} snapshot data.

It turns out that a necessary condition for solving this optimization problem is the invertibility of the matrix A, and this condition is shown to be satisfied [6] if the p^{th} derivative of Φ is completely monotonic in $[0,\infty)$. One choice of such a function is the exponential function; i.e.

$$\Phi_n(\|x_m - c_n\|) = e^{-\frac{\|x_m - c_n\|}{\sigma}} \tag{13}$$

The target height corresponds to the value of h that gives the maximum response in the output $y(h)$.

There is one fundamental decision that needs to be made before the implementation of this technique. That is the determination of the centre vectors. The original RBF technique derived from the function approximation theory uses all of the multidimensional data as centres. However, this will form a huge matrix A which it is impractical to compute. Furthermore, the redundancy brought about by using all data points as centres may lead to misleading variations due to noise data. It was suggested in [7] that the number of centre vectors can be less than that of the input vectors. This results in a simpler non-square matrix. A least squared method can now be used to solve the matrix. The pseudo-inverse of A will provide a solution with the minimum least square errors. This optimum least square solution may be obtained either through direct evaluation of the pseudo-inverse A or through an adaptive algorithm such as the Least Mean Squares (LMS) algorithm or the Recursive QRD Least Squares algorithm [8].

In normal circumstances, the Gaussian nodes in a RBF network uses a constant in the

denominator of the exponent in (13). The constant σ should be determined a priori so that the estimation of the weights can be truly linear. In our case, we consider specific class of RBF networks, called classifying networks, in which a small value is assigned to σ in the exponent of the basis function. The approach of a classifying network is similar to that of the Pazen-window approach [2] which estimates the nearest neighbour of the input. Actually, the classifying network finds the input vectors that are the nearest neighbour of all the centre vectors in the hidden layer according to a distance measure. When a centre vector matches the input vector, the distance function is the smallest and the output of the Gaussian node of the of the RBF network, i.e. $\exp(-\|\cdot\|/\sigma)$, takes on its largest value. Each output represents the degree of closeness between the centre vector and the input vector. Classification is accomplished be finding the maximum among all the outputs. The constant σ serves as a sensitivity factor. If the σ is small, the hidden layer will be more sensitive to the fine changes in the distance function and discriminate against any other centre vector that is further away from the input vector.

3 Computer Simulations

In this section are presented the computer simulation results using synthetic radar signals which are assumed to have propagated through the troposphere with certain characteristics. The simulations were carried out to study (1) the error performance of the technique under various conditions and (2) the tracking performance of the technique for a moving target.

To start with, a simple simulation was conducted to test the algorithm's effectiveness in tracking a stationary target. For comparison purposes, the parameters used in this simulation were set to be exactly the same as those used in the experiments which are to be presented later. This simulation was based on one way propagation between a beacon operating at 10.2 GHz and a 32 element, 1.82 m aperture antenna array, which are 4.61 km apart. The synthetic data were generated based on (1) with SNR = 10 dB. The target height was chosen to be 9 m above the water surface, and one hundred trials were carried out. Then fifteen snapshots were grouped together to form a single target height estimate. Prior to processing, each signal vector was normalized with respect to the amplitude and phase at the top element of the array. Fig. 2 shows a trace of the height estimation. It is clear that the new technique can be used to track a target steadily with a high precision. The height estimates were always within 0.5 m of the actual

target height. The output spectrum in Fig. 3 demonstrates the algorithm's effectiveness in terms of its threshold and the accuracy with which it indicates the target height. The spectrum for the new technique has only one single sharp peak much like a delta function, rather than many ambiguous peaks which characterize the maximum likelihood approach using the same signal model [9].

One of the difficulties in processing the data is that the signal amplitudes and initial phases are unknown. One possible solution is to normalize the signal vectors with respect to the amplitude and phase of one of the array elements. However, it was observed that the performance depended on which element was used as the reference for the normalization. For optimal performance, it is suggested to respectively normalize the input signal vectors with respect to each element of the array and then to jointly process the normalized signal vectors [10]. As a simplified example of the the proposed normalization technique, we employed two references to carry out the normalization.

Three performance curves for the RBF network are shown in Fig. 4. The results corresponded to target heights 9m, 11m, and 15m, respectively. All the parameters were kept the same as before The mean square error (MSE) is calculated as followed

$$\mathcal{E} = \sum_{i=1}^{N_t} \frac{(h-\hat{h}_i)^2}{N_t} \tag{14}$$

where h is the true target height, $\hat{h}_i$ is the i^{th} estimate of h, and N_t is the total number of trials of ensembles to be averaged. In the case shown in Fig. 4, N_t was taken to be 100. Based on these results, the new technique shows promise. The performance threshold accounted for SNR values of less than 7 dB. For the case where h=11 m, it was as low as 5 dB.

We now consider the case where the low flying target is in motion. A simulation based on a single vertical moving target was used to demonstrate the true capability of the new method in tracking a single target over the range of the potential target heights. All the parameters were exactly the same as before, except that the target height was an exponential cosine function of time

$$h(t) = -20e^{\frac{t}{800}}\cos(\frac{\pi t}{600})+25 \qquad (15)$$

There was no particular reason for choosing this function, rather than synthesizing a nonlinear motion for the target. In Fig. 5 are shown the tracking results. The corresponding SNR was set to be 15 dB. Again, 15 snapshots were used to derived an estimate. These 15 snapshots constituted a time window. In other words, a sliding window, consisting of 15 snapshots, was used as the estimates were derived. However, since the target height was a function of time, the 15 snapshots comprising a window corresponded to different heights. The dashed lines in Fig. 5 indicate the heights that corresponded to the first and last snapshots within the time window. It is then expected that the estimates should fall into the height interval marked by the dashed lines. Indeed, the results showed that this was the case.

In Fig. 6 is shown another case where SNR=12 dB. Once more, the tracking performance of the new algorithm was rather commendable, even though an outliner appears in the results. Outliners, however, can be removed by employing a Kalman filter or otherwise.

The advantages brought about by combining the deterministic multipath model with the RBF network are not only limited to high precision tracking at low SNR and the ability to trace a moving target. The simple parallelism behind the method also permits an easy implementation in a parallel processing computer. Besides, it does not require more than one training cycle nor the calculation of the inverse of a large matrix. Therefore, a real time processing is highly possible. The block diagram of the new low angle-tracking algorithm is illustrated in Fig. 7.

4 Real Data Experiment

In late October 1987, a series of experiments were performed on the west coast of the Bruce Peninsula, Ontario, Canada, overlooking Lake Huron. This particular location was chosen because of the high sea states caused by the combination of the westerly winds, the shallow water offshore, and the long fetch across Lake Huron. One of the objectives of these experiments was to acquire data suitable for elevating the performance of high-resolution estimation techniques for low-angle radar tracking applications. A 32-element sampled aperture antenna, developed at the Communications Research Laboratory, McMaster University, Hamilton, Ontario, Canada, was

used to collect multipath data under a variety of meteorological and water surface conditions. The experimental system consisted of a transmitter, a sampled aperture antenna, and a data acquisition system. The transmitter was a 100-mW CW beacon, sited at a distance of 4610 metres from the receiving antenna. The height of the transmitting horn could be varied through the height interval from 3.5 to 18.5 metres above the water level. The receiving antenna array has a 1.82-metre aperture and consists of 32 10-dB H-polarized horns. It follows from the geometry that the angular separation of the direct and reflected signals varied from 9% to 50% of a BW. All elements in the array were precisely aligned and uniformly distributed to within 0.1 mm to minimize errors due to placement of the antennas. The height of the centre of the sampled aperture antenna was fixed at 8.8 metres above the water level. The data were recorded with 12-bit precision at a sampling rate of 62.5 samples/second. The system operated at X-band.

Two sets of data were used to test the new algorithm. Each of the data sets contained 127 snapshots. The averaged amplitude and phase of the first data set were plotted along with those of synthetic data in Fig. 8 and Fig.4.9, respectively. The null which appears in the amplitude trace is due to the destructive interference between the direct and indirect signals. In this case, the best fit to the experimental data occurred for a the transmitter height of 9 m above the water surface and a sea wave-height-variation of 0.5 m. However, a large phase offset was observed between the synthetic signal and the measured signals. This difference is plotted in Fig 10. A linear relationship between the phase discrepancy and the location of the array element is observed. Thus, the error may be due to an imperfect calibration of the array or a tilt of the receiving array. In fact, the tilt meter that was mounted in the array was malfunctioned during the experiment. Therefore, the degree to which the array was tilted was unknown. Based on Fig. 10, it is speculated that the array was tilted upward. As well, the phase deviation observed for the 19^{th} element is another source of error. The defect may have been caused by a hardware failure for the 19^{th} element. It should be pointed out that these two errors were also present in a later data set.

For purposes of comparison, the MUSIC algorithm was also used to analyze the data. As mentioned before, the high correlation between the direct and the indirect signals will cause the MUSIC algorithm to fail in resolving the two incoming signals. In an attempt to reduce the correlation, the spatial smoothing technique was used with the array divided into ten subarrays.

The estimation results obtained using the MUSIC algorithm were shown in Fig. 11. Based on the experimental setup, the angular separation between the direct and indirect signals was approximately 0.4°. Therefore, it is easy to see that the MUSIC algorithm failed to resolve the two signals. There were two possible reasons for its failure. First, it was observed that the eigenvalue spread for the covariance matrix was large. This implies that a high degree of correlation still existed between the two signals after spatial smoothing. Second, the fact that the noise background was coloured degraded the MUSIC algorithm's resolving ability.

The new method, based on a deterministic multipath model and a nonlinear network, was used to process the data. Since we know a priori the two major sources of errors, the processing was carried out with and without corrections being made to the data. In Fig. 12 are shown three traces of tracking results. The first trace (the dotted line) was obtained without making any corrections to the data. The second trace (the dashed line) was obtained by omitting the signal at the 19th element. The last trace (the solid line) was obtained by not only omitting the signal at the 19th element, but also by adding a phase term to the data in order to compensate for the tilt of the array. In all three cases, the phase reference was chosen to be the 1st element and fifteen snapshots were grouped together to form a height estimate. Even though the first two cases contained corrupted data, the target was tracked to within a small range in height except at the end of the trace when the phase error of the 19th element became evident. The mean square tracking errors are 5.68 dB and 5.67 dB, respectively. The main source of this error was the false tracking (i.e. the trace jumping to another track) that occurred near the end of this result. Comparing these results with those obtained with the MUSIC algorithm, the nonlinear network demonstrated its superiority in providing a highly accurate angle-of-arrival estimates. The superiority was due to the fact that the new technique did not depend solely on the phase delay between the antenna elements. It also used both the amplitude and phase of the received signal. The first trace showed how the new technique has immunity to the phase error at the 19th element. In fact, the second trace was similar to the first one. Therefore, the major source of error in the estimates was due to the tilted array. The third experiment was successful in tracking the stationary target. The mean square error was -5.4 dB and the maximum error is 1 m.

Another set of measured data was also used to test the performance of the network in the same way. The target height for this data set was 15 m. The amplitude and the phase of the data were

compared with those of the synthetic data, as shown in Figs. 13 and 14. The phase difference between the measured and synthetic signals again exhibited a linear relationship with the array element, as shown in Fig. 15. The MUSIC algorithm, together with spatial smoothing using ten subarrays, was once again used to analyze the data and once again failed, as shown in Fig. 16. Similar to the previous set of measurement, the RBF network is able to provide a accurate estimate of the target height. These results are shown in Fig. 17. In the first trace (the dotted line), a few estimates were outliners. Since they did not appear in the second trace (the dashed line), which was obtained by omitting the signal from the false element, it was likely that the false estimates which appeared in the first trace were due to the phase error associated with this element. In the second trace, the height estimates fell into the neighbourhood of the true target height with a slightly larger bias than was the case for the third trace (the solid line). It seams clear that this small error was caused by tilt of the received antenna. For the solid-line trace, the means square error was -13.2 dB and the maximum discrepancy in target height was 0.2 m. Compared with the former data set, the height estimates in the second data set were less influenced by the phase error which was due to a defect element and dependant more on the amplitude of the signal. This may be related to the height of the target. When the target was located at 9 m, the incoming angle of the direct signal is slightly above the boresight of the antenna and the phase delay between the elements is almost negligible. Thus, it was more sensitive to changes of the phase delay.

The results derived from real-life data clearly demonstrate that the RBF network not only has the ability to track low flying targets under high sea-state conditions and correlated noise environments, but it is also less susceptible to other forms of errors such as phase errors due to imperfections in the antenna array. Furthermore, the new method offers very high performance in terms of resolution accuracy. In the two experiments, the largest deviation was less than one meter. Moreover, the performance of the new method was demonstrated to be robust in the sense that it did not depend on the nature of noise. This desirable attribute is due to the distance function used inside the radial basis function. The output of the RBF depended mainly on the Euclidean distance from the centre vectors. Therefore, it is insensitive to the characteristics of the noise.

5 Conclusions

In this paper, a new technique, based on a radial basis function neural network, has been developed as a solution to the low-angle tracking problem. Computer simulations were carried out to evaluate the performance of the algorithm. It has been shown, using simulated data, that the new technique was able to track both stationary and moving targets under low signal-to-noise ratio environments with high accuracy. It has also been demonstrated, using real-life multipath data, that the new technique is robust in the sense that it was its insensitive to array errors and the nature of the background noise. The algorithm is superior to the MUSIC algorithm in terms of robustness, resolvability, and estimation accuracy. On the other hand, more study is needed to understand how other factors, such as the centre vectors and coefficients used in the basis functions, affect the performance of the algorithm.

References

[1] E. Jakeman and R. J. A. Tough, "Statistical models of low-angle forward-scattering of microwaves," IEE conf. Publ. 281, Radar-87, pp 568-573, 1987.

[2] T.Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol.78, no.9, pp. 1481-1496, 1990

[3] R. P. Lippman, "An introduction to neural net," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[4] A. S. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modeling," Technical Report LA-UR-87, Los Alamos National Laboratory, 1987.

[5] T. Lo, J. Litva, "Low-angle tracking using multifrequency sampled aperture radar," *IEEE Trans. Aerospace and Electronic Systems*, vol. 27, pp.797-805, Sept. 1991.

[6] C. A. Miccelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Const. Approx.* vol. 2, pp. 11-22, 1986.

[7] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex System*, vol. 2, pp. 321-355, 1988.

[8] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, 1991.

[9] E. Bosse, R. M. Turner, and M. Lecours, "Tracking swerling fluctuating targets at low altitude over the sea," *IEEE Trans. Aerospace and Electronic Systems*, vol. 27, pp.806-827, Sept. 1991.

[10] T. Wong, "Low-angle target tracking based on a deterministic multipath model and radial basis functions," M. Eng. Thesis, McMaster University, 1991.

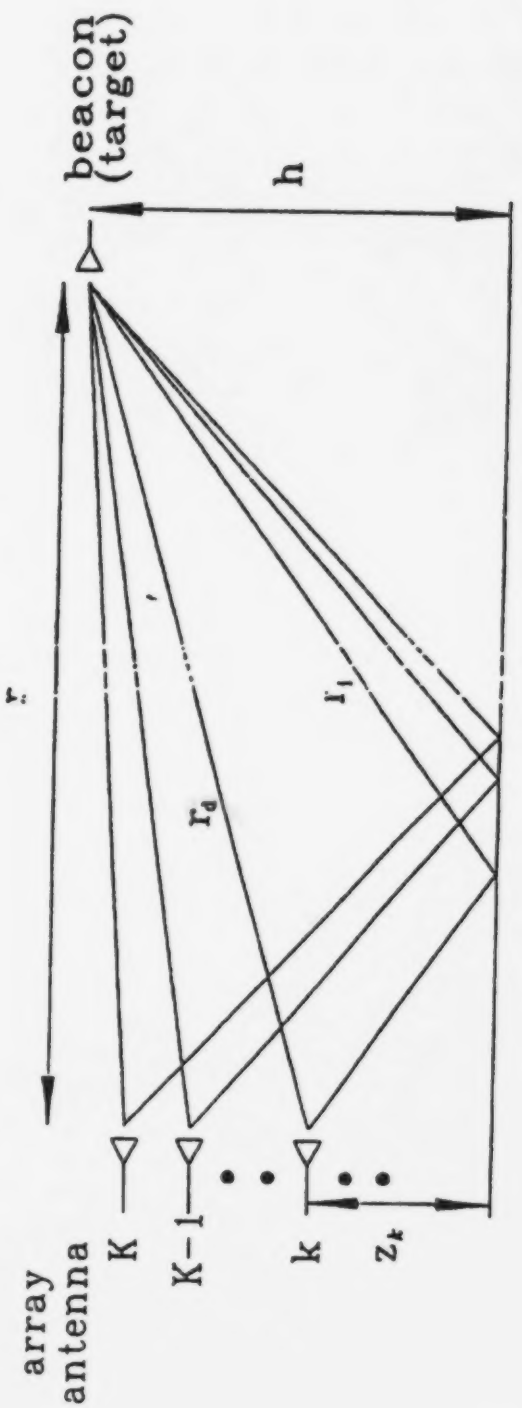


Fig. 1 Geometry of a simple flat-earth model for multipath.

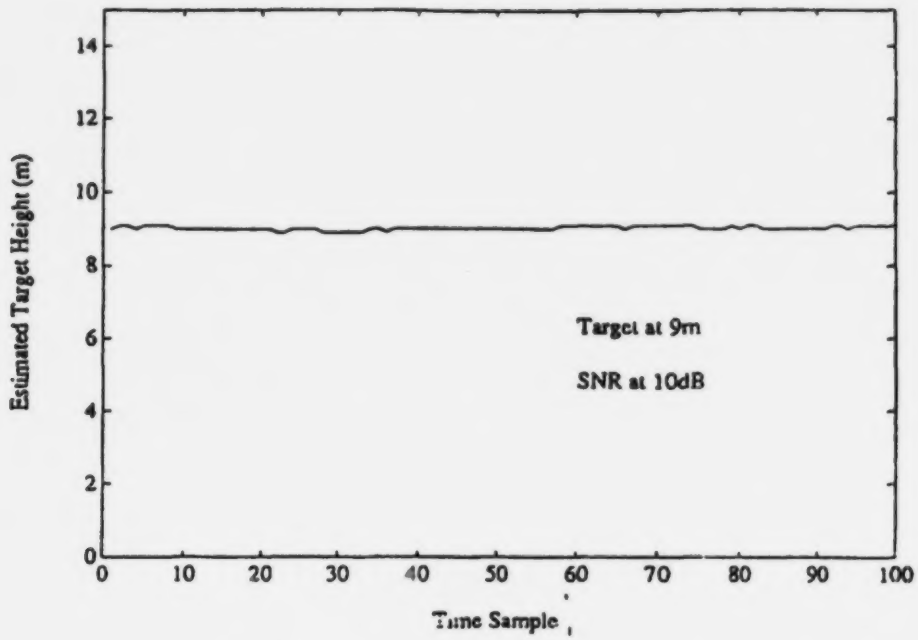


Fig. 2 A trace of target height estimates.

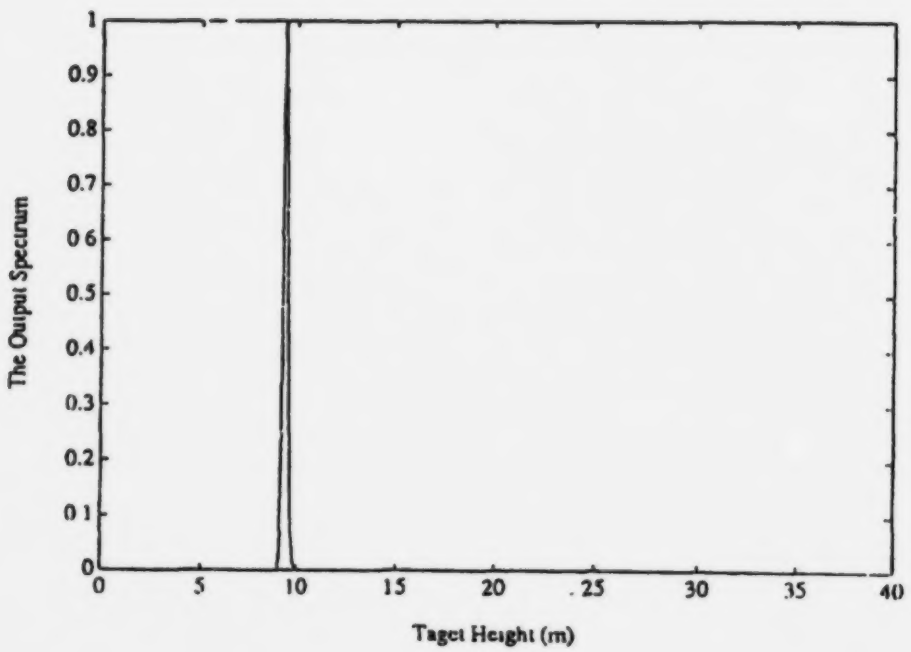


Fig. 3 The output spectrum of the RBF network.

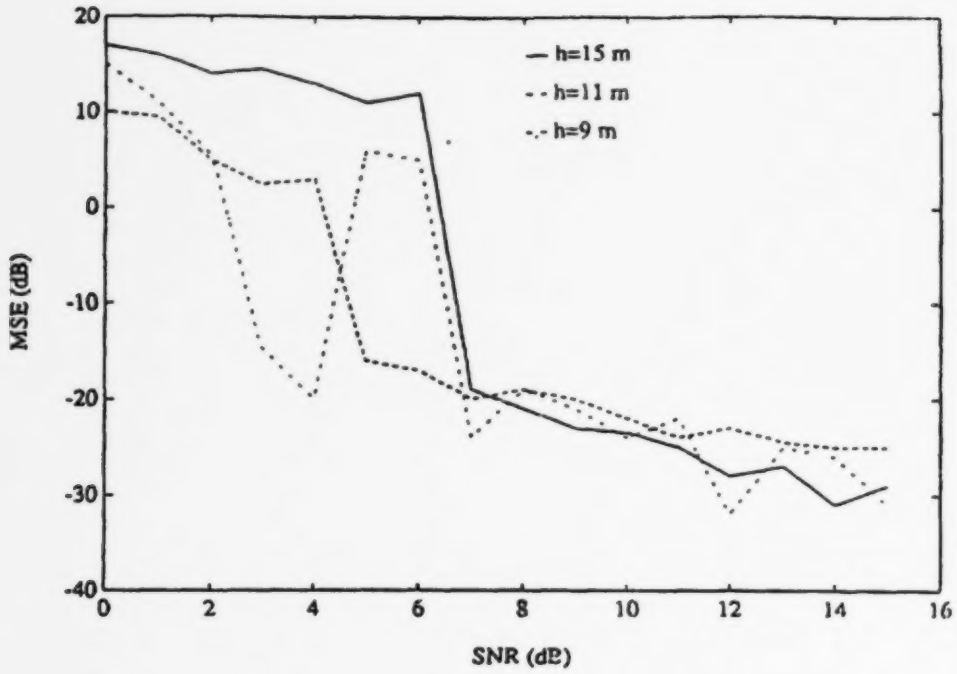


Fig. 4 The performance curves for the RBF network.

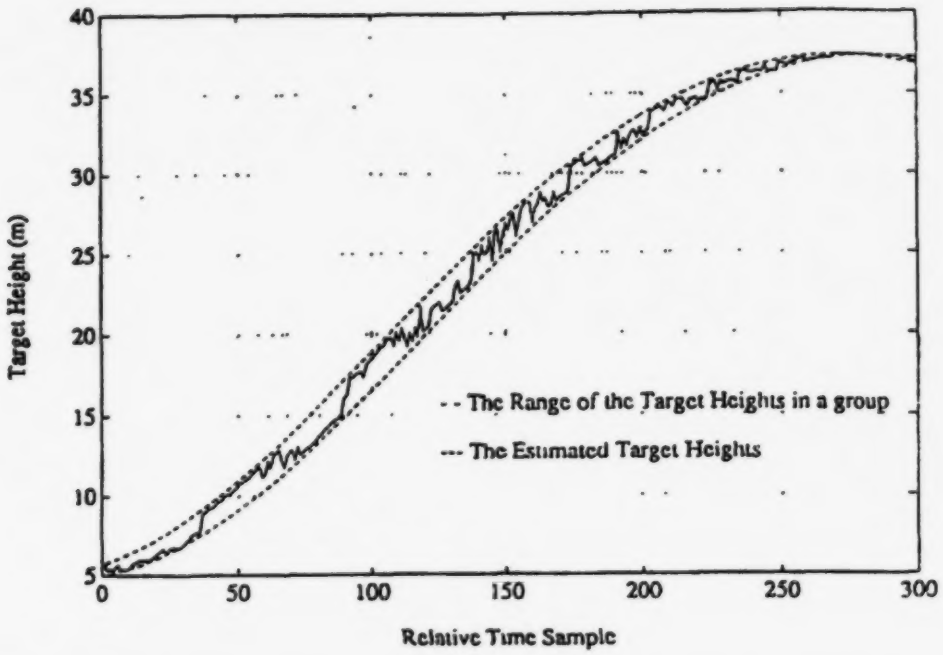


Fig. 5 The tracking results for a moving target; SNR=15 dB.

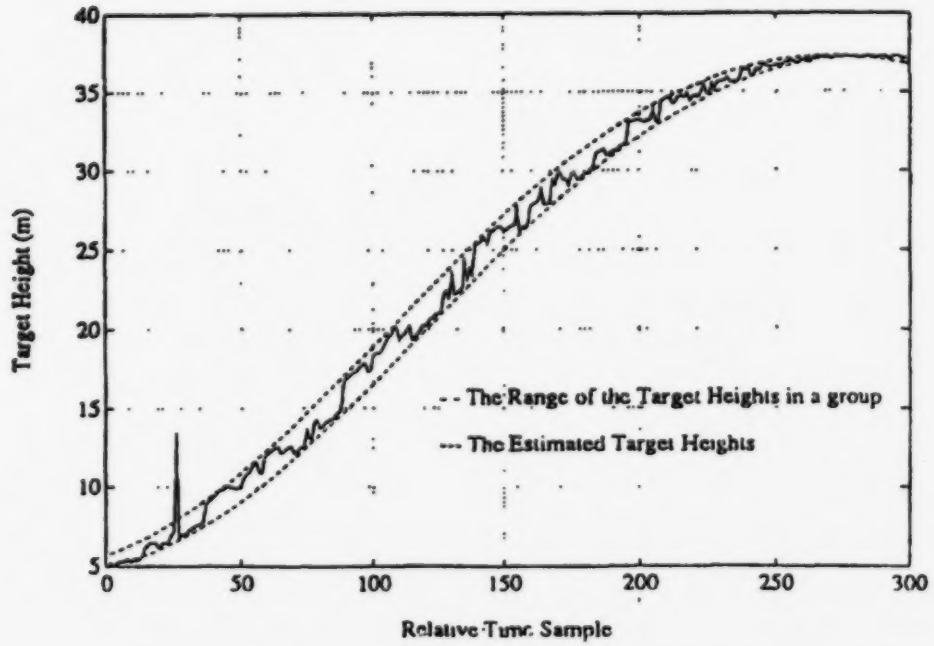


Fig. 6 The tracking results for a moving target; SNR=12 dB.

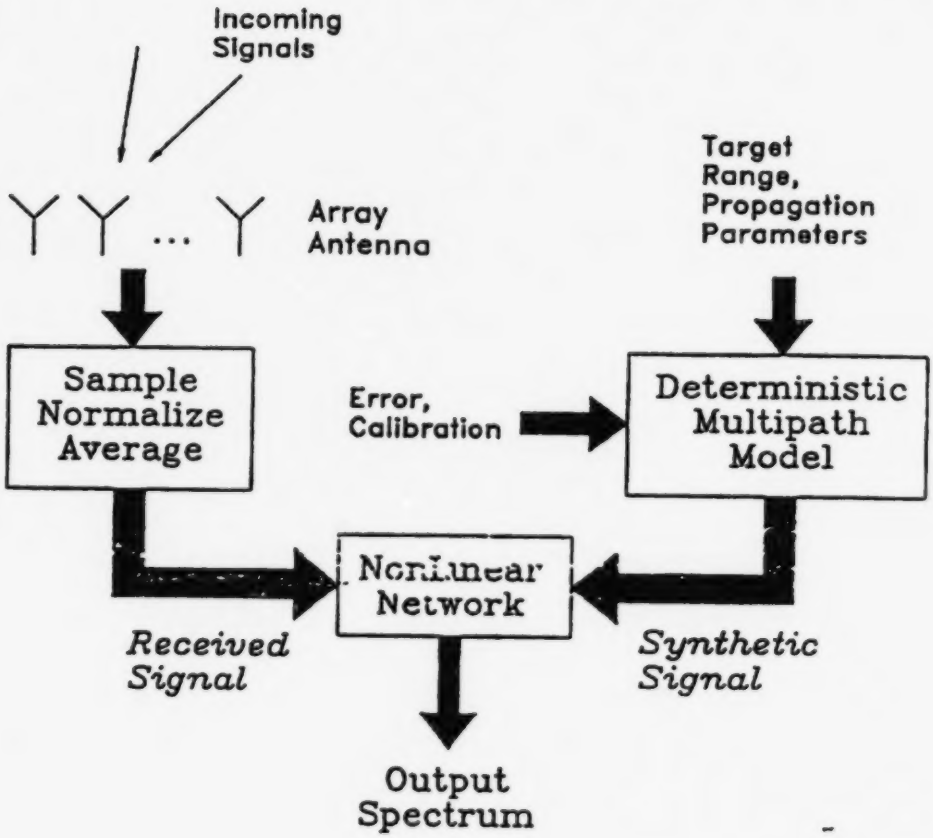


Fig. 7 Functional system diagram for implementing the RBF network.

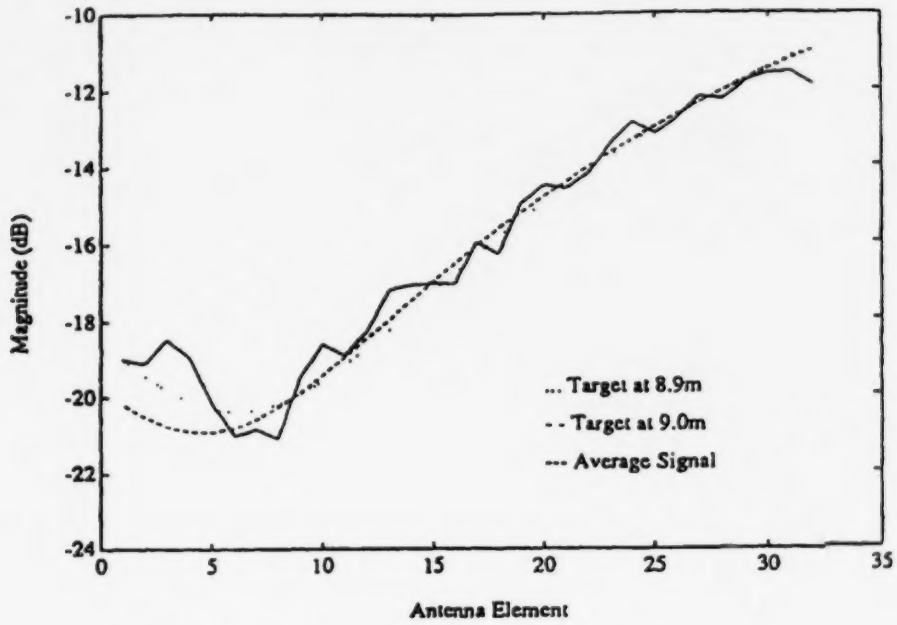


Fig. 8 Comparison of the amplitude distributions of synthetic data and the first set of the measured data along the array face

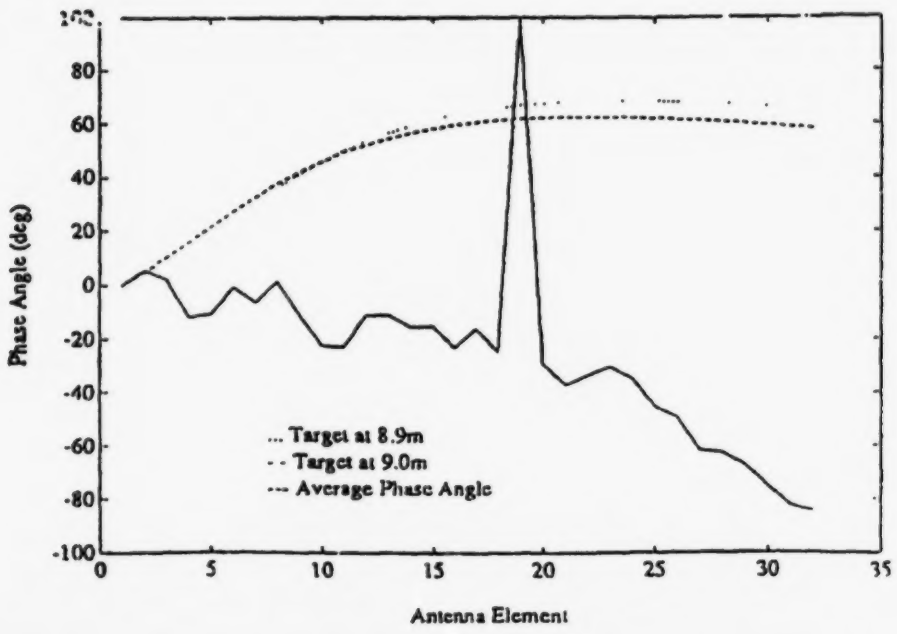


Fig. 9 Comparison of the phase distributions of synthetic data and the first set of the measured data along the array face.

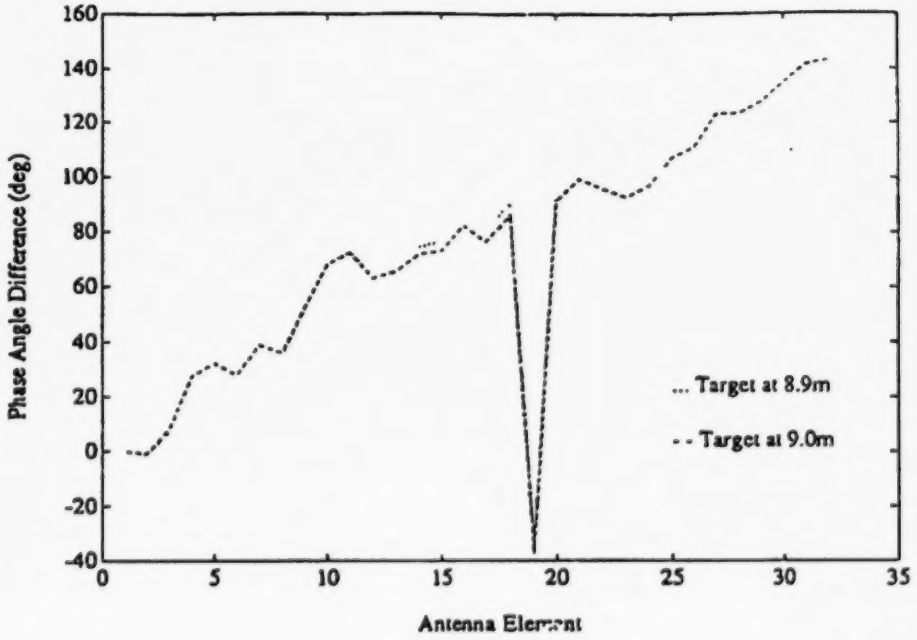


Fig. 10 Phase differences between the first data set and synthetic data.

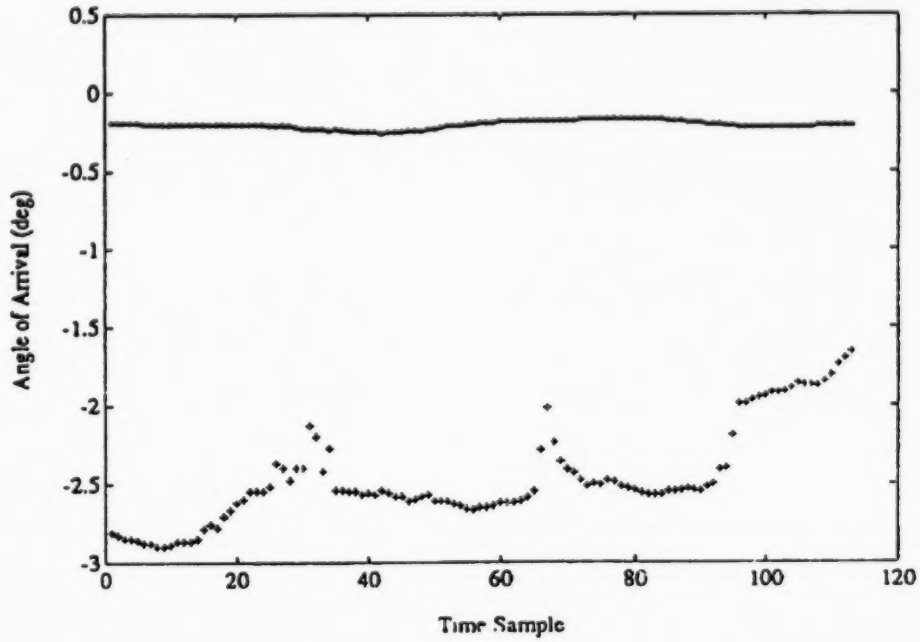


Fig. 11 The AOA results derived using the MUSIC algorithm.

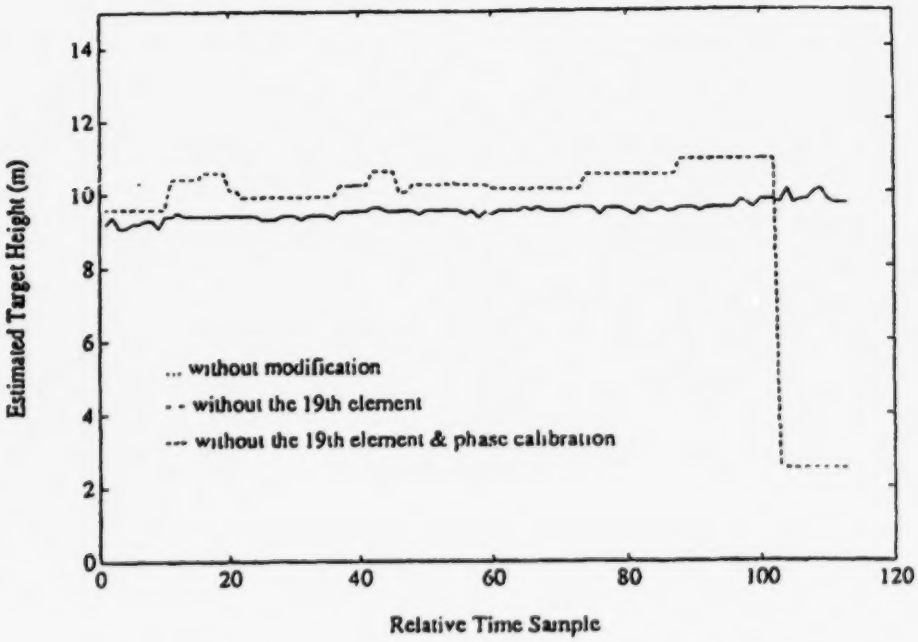


Fig. 12 The target height estimates obtained using the RBF network.

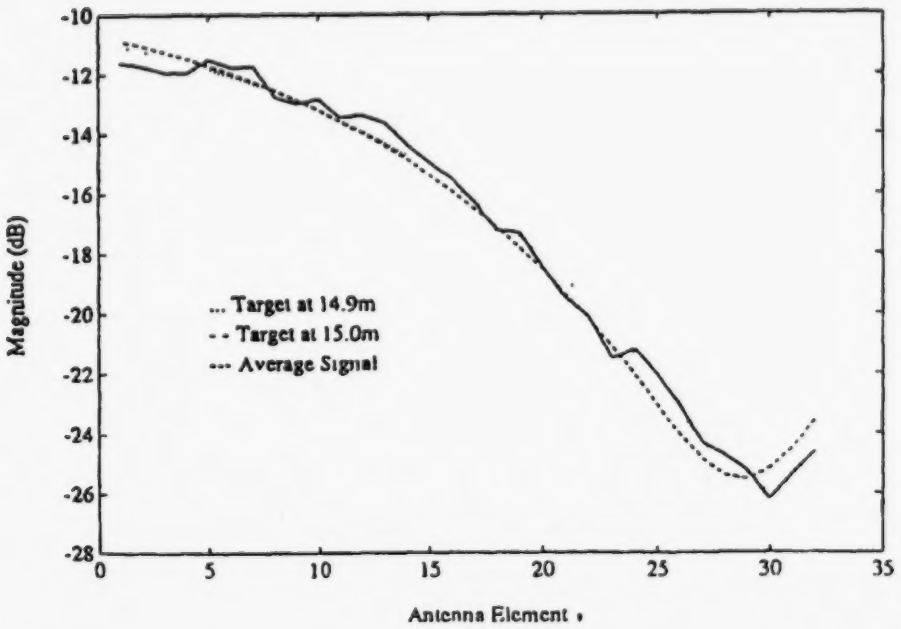


Fig. 13 Comparison of the phase distributions of synthetic data and the first set of the measured data along the array face

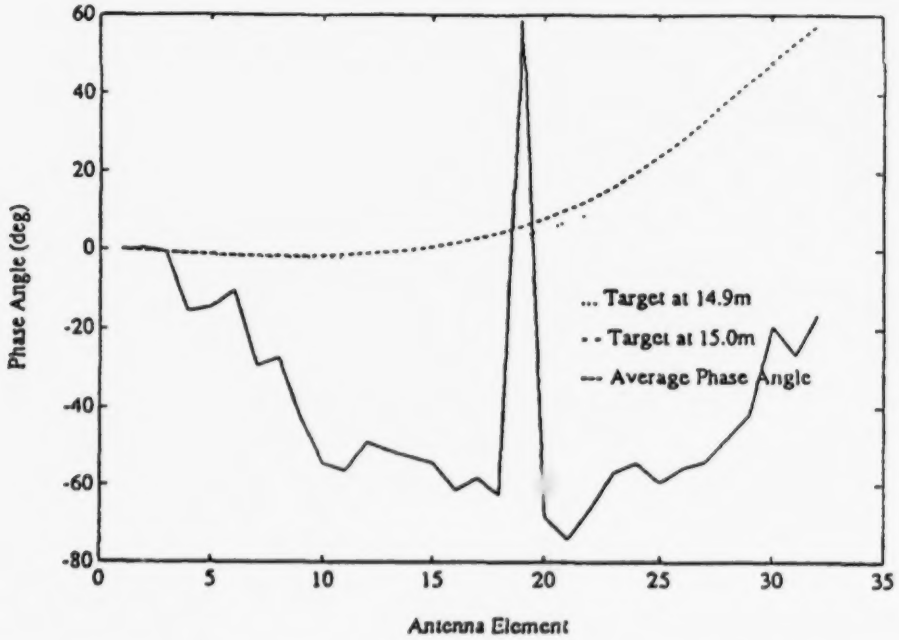


Fig. 14 Comparison of the phase distributions of synthetic data and the first set of the measured data along the array face.

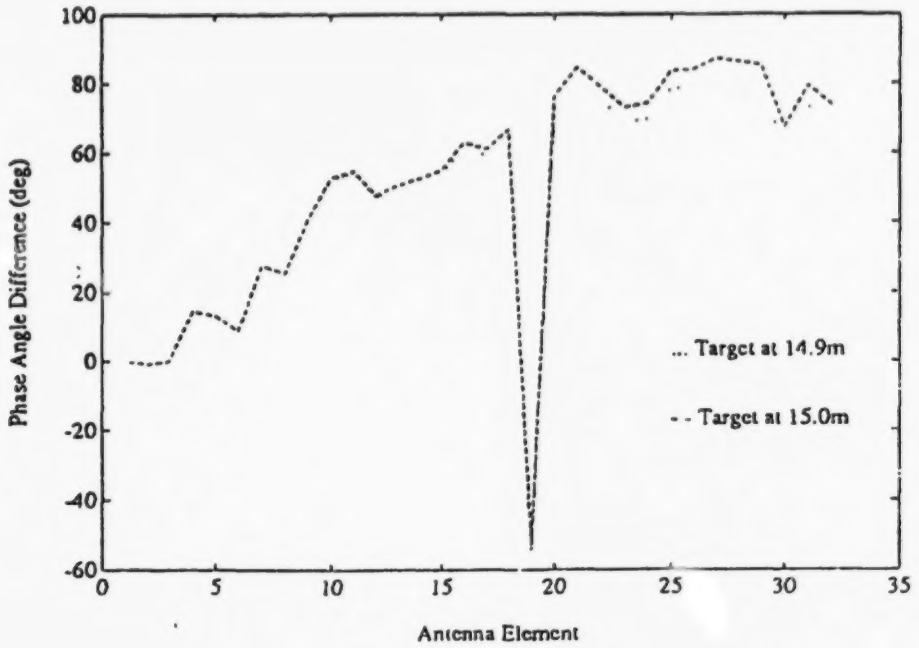


Fig. 15 Phase differences between the first data set and synthetic data.

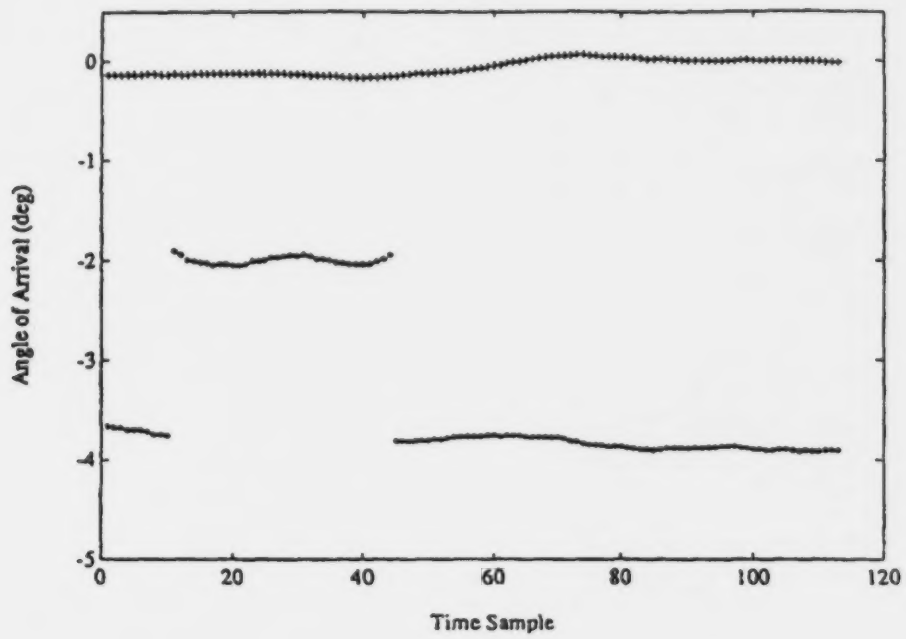


Fig. 16 The AOA results derived using the MUSIC algorithm.

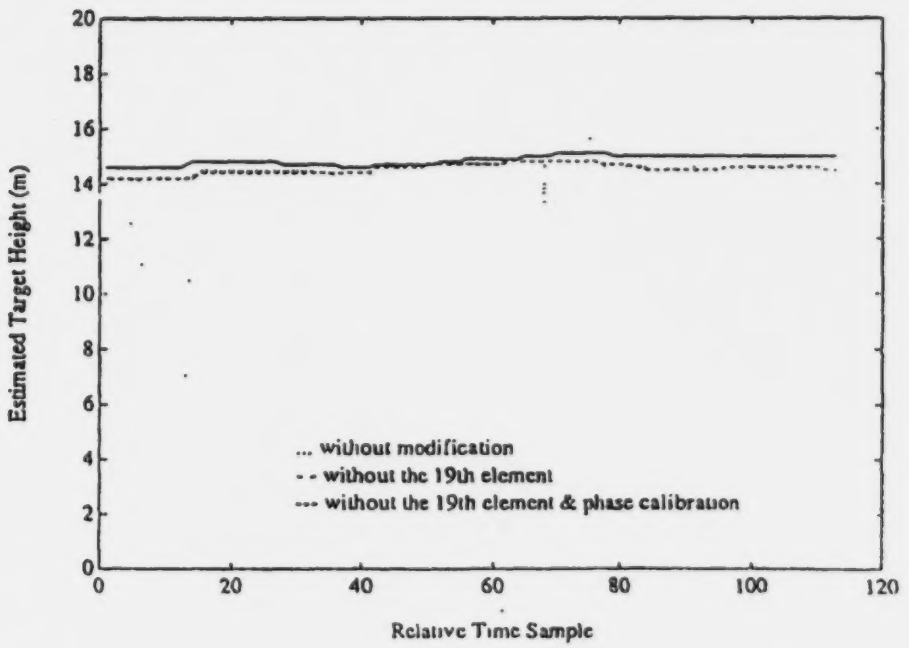


Fig. 17 The target height estimates obtained using the RBF network.

Some Neural Computing Approaches to Target Tracking

by

Joseph N. Maksym

Defence Research Establishment Atlantic

Abstract

Multiple targets, false detections, and manoeuvres present difficulties in the application of tracking and data association algorithms. The discussion examines potential applications of neural networks in target tracking and data association problems. While neural computing has proved useful for many pattern recognition and control applications, it is not yet clear how and where neural networks should be used in systems for tracking targets. Preliminary work on the application of neural networks as adjuncts to more conventional Kalman filter tracking systems will be discussed. This paper examines the possible use of neural networks applied to model selection in a multiple model approach to tracking manoeuvring targets, as well as a neural network approach to measurement-to-track data association.

Overview

- Kalman Filter Target Tracking
  - Signal model
  - Kalman equations
  - Manoeuvres
  - Data Association
- Neural Networks
  - Optimization networks (Hopfield)
  - Rapid learning (Gaussian Sum)
  - Multisensor track initiation

NOTES

The talk is in two parts.

The first part is a kind of introduction to some of the issues involved in target tracking. It reviews a traditional approach – target tracking with Kalman filters.

Some of the issues involved are signal models, the Kalman equations, the effect of manoeuvres, and the problem of data association.

The second part looks at several neural computing paradigms that may have application to target tracking – especially for multiple target multiple sensor situations.

In particular we will examine Hopfield networks, and rapid learning networks applied to a multisensor track initiation problem.

Multiple Target Stochastic Model

$$\mathbf{x}^t(k+1) = \mathbf{F}^t\,\mathbf{x}^t(k) + \mathbf{G}^t\,\mathbf{u}^t(k) + \mathbf{f}^t(k+1|k)$$

$$\mathbf{z}_j(k) = \mathbf{H}_j\mathbf{x}^t(k) + \mathbf{w}_j(k)$$

Time index: k = 0, 1, ... , N

Target index: t = 1, 2, ... ,S

Measurement index: j = 1, 2, ... , m(k)

NOTES

First a review of the Kalman filter approach to multiple target tracking.

This is basically a parametric method. We model the target state (e.g. its position, velocity etc.) by the top equation. In general there is one such stochastic model for each target.

u(k) is a zero-mean white Gaussian noise vector with covariance **Q**, and f(k+1|k) represents a deterministic input such as a target manoeuvre, and w(k) is a zero-mean white Gaussian noise vector with covariance R.

At each time step k a number m(k) of measurements are made. This number will often differ from the number of targets because of missed detections and false alarms.

Kalman Filter State Estimation

Measurement Update* :

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\,[z(k) - H\,\hat{x}(k|k-1)]$$

$$S(k) = H\,P(k|k-1)\,H^T + R$$

$$K(k) = P(k|k-1)\,H^T\,S(k)^{-1}$$

$$P(k|k)^{-1} = P(k|k-1)^{-1} + H^T\,R^{-1}\,H$$

Time Update* :

$$\hat{x}(k+1|k) = F\,\hat{x}(k|k)$$

$$P(k+1|k) = F\,P(k|k)\,F^T + G\,Q\,G^T$$

*Target index t has been dropped for clarity.

Defence Research Establishment Atlantic

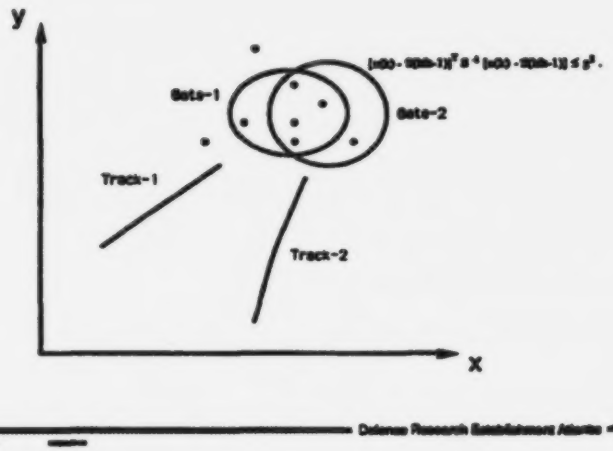
NOTES

The basic Kalman filter equations are summarized here (the target index t has been dropped).

Basically, Kalman tracking consists in picking an initial state for each target, and an initial value for the filter covariance matrix P, and then updating these expressions using measurements made at each time step.

There are two steps: a measurement update where the target state and covariance estimates are updated to account for the new measurement, and a time update, which uses the assumed target model to extrapolate to the next time step.

Data Association Problem



NOTES

Multiple targets complicate the problem of data association, as illustrated here for a simple situation with two converging tracks.

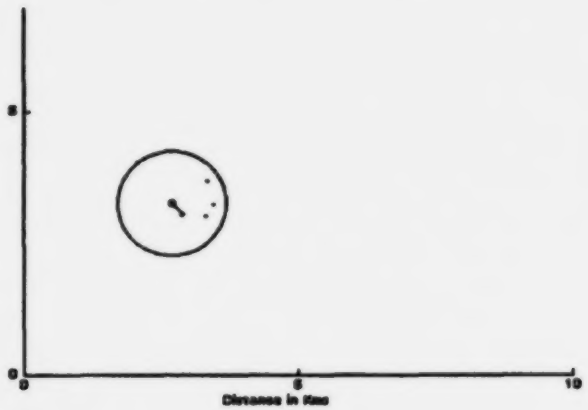
The problem is to decide which observation is to be associated with each track.

Each Kalman filter computes a gate region centred on the expected measurement (Size of the gates depends upon the target stochastic model as well as measurement noise).

Normally the gate size is chosen such that the true target return will be inside the gate, say 0.999 of the time.

A very simple data association techniques is the nearest neighbour method, which minimizes the distance of the chosen measurement from the prediction made by the filter. The distance measure illustrated in the viewgraph can be used.

Nearest Neighbour: k = 3

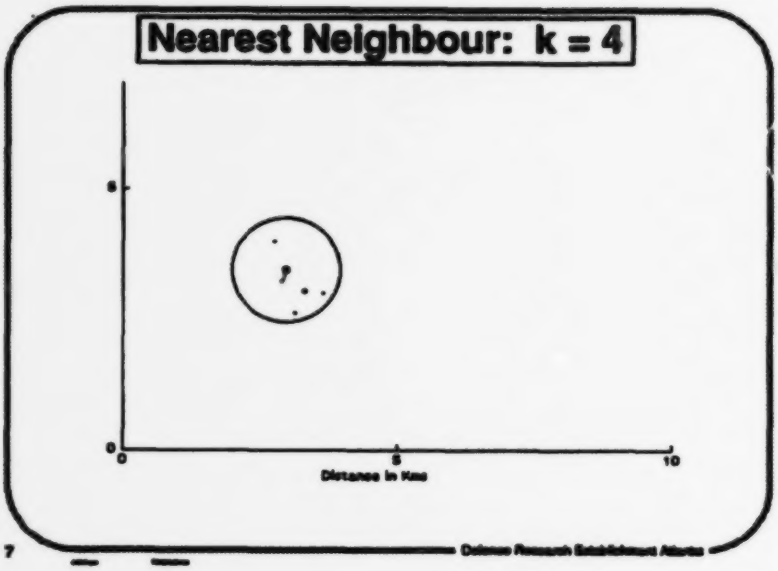


NOTES

This viewgraph illustrates the nearest neighbour method in a simulation.

At time = 3 we have three false alarms (small dots) and the true return inside the gate.

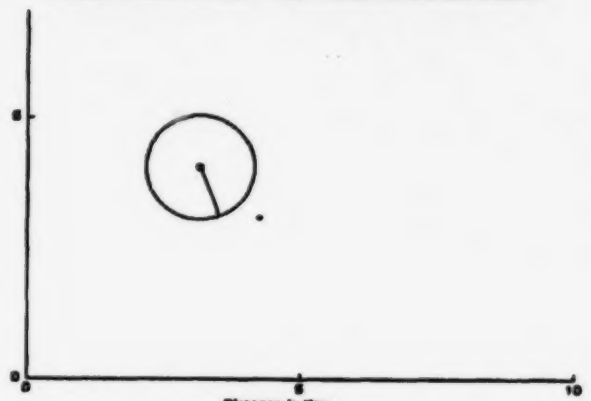
The nearest-neighbour associated the true return with the track.



NOTES

At time = 4 the nearest measurement is a false alarm.

Nearest Neighbour: k = 6



NOTES

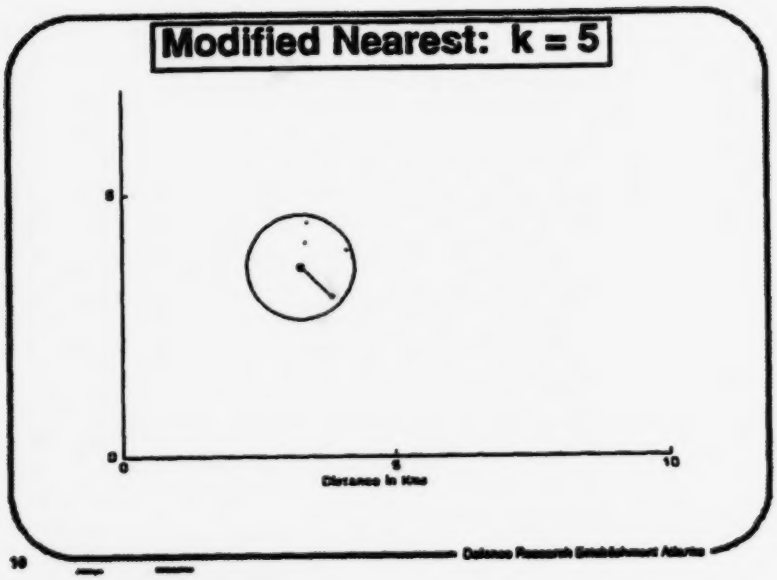
At time = 6 the filter loses the true return – it falls outside the gate.

Tracking a Manoeuvring Target

NOTES

Clearly, a Kalman filter based on assumed dynamics (constant velocity target, for example) is mismatched when a strong manoeuvre occurs.

This is shown here. A good deal of work on adaptive filters is reported in the literature on Kalman filter tracking. Most of these adaptive studies propose some form of system identification, which models the non-white residual error occurring during a manoeuvre.

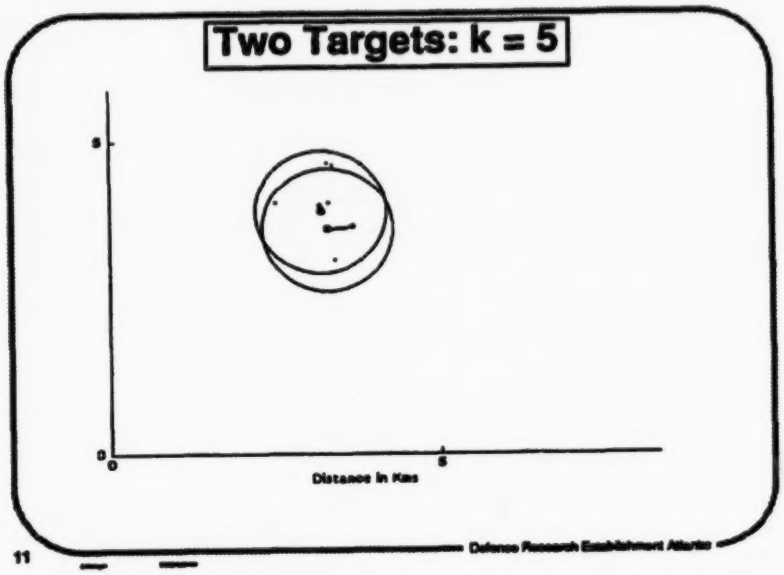


NOTES

Basically, when a residual associated with a measurement is highly correlated with past residuals the measurement is a good choice for association.

A simple technique for improving nearest neighbour data association can be based on this concept.

The viewgraph illustrates how the modified nearest neighbour tends to associate the true return rather than a false alarm, which happens to be closer to the expected measurement.

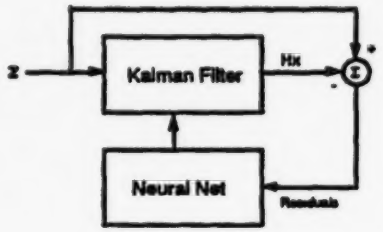


NOTES

In this viewgraph an example with two crossing targets is shown.

The modified nearest neighbour association can assist in preventing target returns from being associated with the wrong track.

Manoeuvre Identification



12 Defence Research Establishment Atlantic

NOTES

The examples we have just seen indicate the importance of having a correct model for a target – during manoeuvres the Kalman filter becomes mismatched.

The use of neural networks for system identification is now well established.

The viewgraph indicates, conceptually, how it might work in a Kalman filter tracker. Here the neural net would be used as an adjunct to a Kalman filter.

There are several problems, however:

1. Standard backpropagation networks have long training times and would need to be trained off-line.
2. A large variety of manoeuvres are possible.

Neural Networks

Some Network Types

- Multi-layer Backpropagation
- Hopfield Networks (energy minimisation)
- Self-Organizing Nets (Kohonen, ART)
- Probabilistic Networks

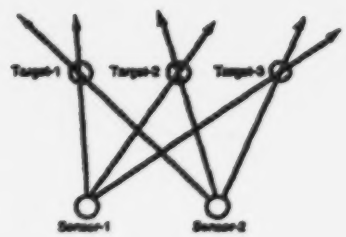
NOTES

The remainder of the talk will look at some neural network concepts for multiple sensor target tracking.

Some of the neural architectures considered were the multilayer backpropagation network, energy minimisation networks such as the Hopfield, self-organizing nets, and probabilistic networks.

The discussion will focus on Hopfield and Probabilistic networks.

Multi-sensor Multi-target Tracking



14

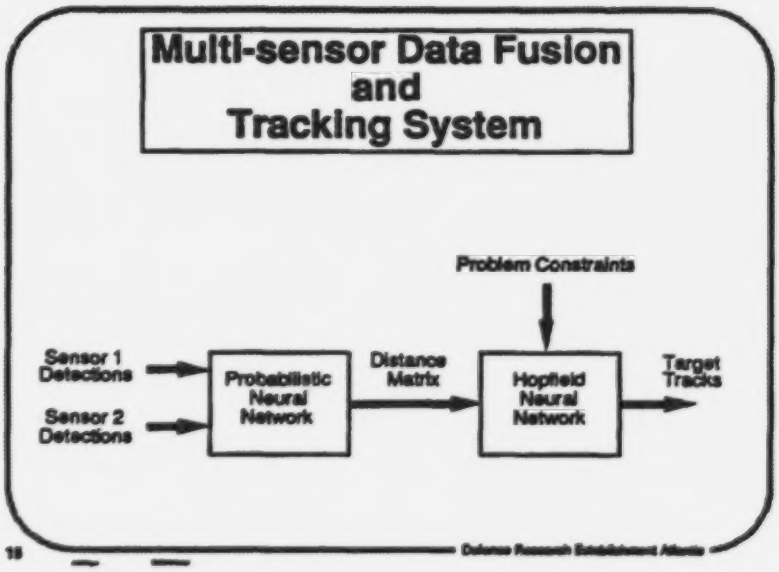
NOTES

Consider a scenario consisting of three targets and two sensors that produce only a bearing indication for each target. In practice, noise will introduce bearing errors, false detections, and missed detections, but these issues will be ignored for the moment.

Given the set of detected bearings at each sensor the tracking problem is to determine the location of the three targets.

There are two aspects of the problem: data fusion, and constrained optimization.

We will examine a system that combines two neural architectures, a probabilistic network and a Hopfield network.



NOTES

The inputs to the probabilistic network are the sets of detections from the sensors.

If a detection at one sensor and a detection at another sensor are similar the probabilistic network produces a high output – that is the distance measure between feature vectors is small.

The resulting distance matrix is passed to a Hopfield network where it determines, in part, the interconnection weights.

The Hopfield interconnection weights are also influenced by problem constraints.

Problem constraints might include a limit of one target present on each detected bearing, an impossibility of targets in a given region, etc.

Probabilistic Neural Network

$$\Omega = \{\omega_1, \omega_2, \ldots \omega_s\}$$

$$p(x \mid \omega) = \frac{1}{(2\pi)^{\frac{n}{2}} \sigma^n m} \sum_{k=1}^{m} \exp\left[-\frac{\|x - x_{ik}\|^2}{2\sigma^2}\right],$$

$$g_i(x) = p(x \mid \omega_i)\, p(\omega_i), \qquad i = 1, 2, \ldots, S$$

16

NOTES

This briefly summarizes a probabilistic network (It may also be referred to as a Gaussian sum or Parzen window approach).

Suppose we have S classes of pattern, and that within each class there is some variation, although smaller than variation between classes.

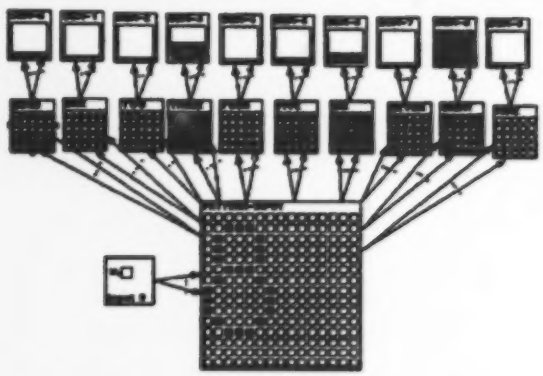
We can develop a density function for the feature vector x for a class as the summation of some number, say m, of individual Gaussian densities, each centred on an example feature vector.

The resulting "Gaussian sum" thus reflects the (variations) within the class.

Training the network is rapid – a set of network weights are stored for each training exemplar.

After training, the network computes discriminants $g_i(x)$ for each class.

Character-Recognition Network



17

NOTES

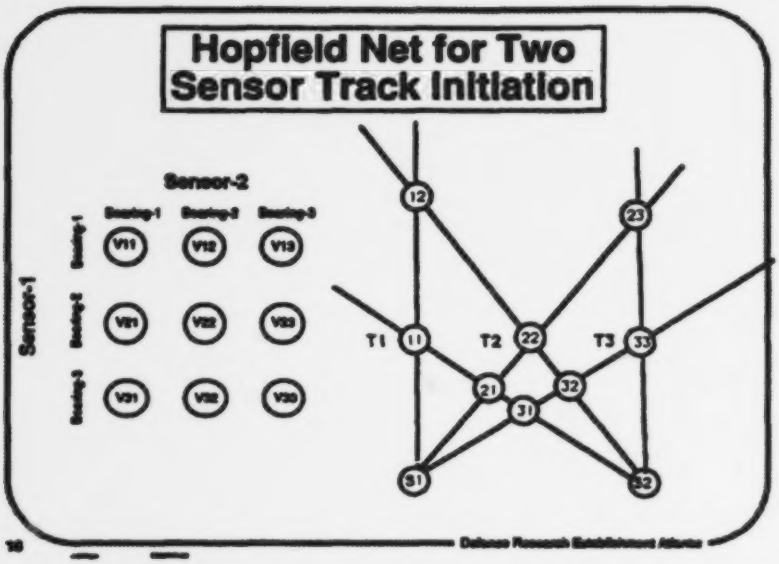
A character-recognition example illustrates the implementation.

An square array of neural units receives input characters.

There is a second layer of units for each class. The links from each unit in the second layer store example patterns during the learning phase. For this network there are twenty-five units in the second layer for each class. Consequently, up to twenty-five exemplars can be stored.

The top layer is a set of single units. Each unit forms the Gaussian sum. The output values (height of black bar) indicate probability of the input belonging to the various classes.

Here, the trained network is presented the character "eight". The largest output is that corresponding to the class "eight", although some output is observed for the classes "three" and "six" due to pattern similarities.



NOTES

The viewgraph illustrates a mapping between potential target locations and neurons in a two-dimensional Hopfield net.

The target location T1, for, example maps into the neuron labelled v11.

With three detections on each sensor there will be nine potential target locations consistent with the observed detections.

Network Design

Cost function:

$$E = \frac{A}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{\substack{k=1 \\ k\neq j}}^{3} v_{ij}v_{ik} + \frac{B}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{\substack{l=1 \\ l\neq i}}^{3} v_{ij}v_{lj} + \frac{C}{2}\sum_{i=1}^{3}\sum_{j=1}^{3} d_{ij}v_{ij}$$

Minimisation favours the following:

- Network states with at most one "on" neuron per row.
- Network states with at most one "on" neuron per column.
- Network states where "on" neurons correspond to a low distance measure d_{ij} between detection-i on sensor-1 and detection-j on sensor-2.

NOTES

Hopfield network design is usually based on definition of an energy or cost function that the network is expected to minimize.

Shown here is a possible cost function for the two-sensor three-target problem.

Basically, the network is to select those target locations that have maximum distance of target features subject to satisfying constraints of the problem.

In most applications of the Hopfield net seen in the literature, constants A, B, C, etc. were empirically adjusted for good performance. This is seen here as potential source of difficulty.

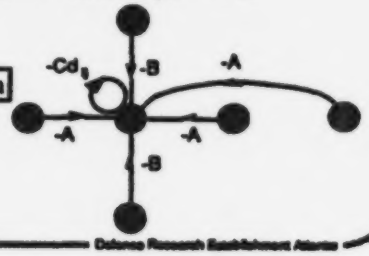
Interconnections

$$E = \frac{A}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{\substack{k=1 \\ k\neq j}}^{3} v_{ij}v_{ik} + \frac{B}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{\substack{l=1 \\ l\neq i}}^{3} v_{ij}v_{lj} + \frac{C}{2}\sum_{i=1}^{3}\sum_{j=1}^{3} d_{ij}v_{ij}$$

Calculation of T_{ijkl}

$$T_{ijkl} = -A\,\delta_{jl}\left(1 - \delta_{ik}\right) - B\,\delta_{ik}\left(1 - \delta_{jl}\right) - C\delta_{ik}$$

Connections into the ijth neuron



Defence Research Establishment Atlantic

NOTES

The form of the energy function translates into link weights.

This is shown in the sketch, which indicates links going into the ij-th neuron.

First, there are inhibitory links of strength A acting along each row.

Next, there are inhibitory links of strength B acting along each column.

Finally, each neuron has a cyclic link (from its output back to its input). The strength of this link is dependent on distance – a high distance.

Difference Equations

$$v_{ij} = g(u_{ij}) = \frac{1}{2}\left(1 + \tanh(\lambda u_{ij})\right)$$

$$u_{ij}(t+1) = u_{ij}(t) + \Delta u_{ij}$$

$$\Delta u_{ij} = \left(-\frac{u_{ij}}{\tau} + \sum_{k=1}^{3}\sum_{l=1}^{3} T_{ijkl}\, v_{kl} + I_{ij}\right)$$

NOTES

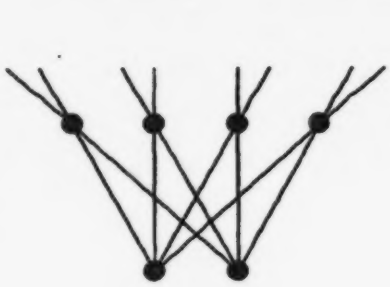
This viewgraph shows the difference equations for updating the network.

Each neuron produces an output v(i,j) that is a nonlinear function of its input u(i,j). The nonlinearity is usually a soft limiter, such as the expression at the top of the viewgraph. Lambda is a gain constant, which determines the shape of the nonlinearity.

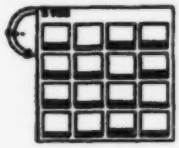
At each iteration the input to a neuron is the sum of: the previous input decayed with a time constant, a weighted sum of the outputs of all neurons in the network, and an input current I.

Actually, Tank and Hopfield's original papers described an analogue implementation involving operational amplifiers, resistors, and capacitors.

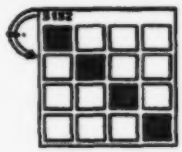
Two Sensor Four Target Example



Example Scenario



Initial network state



Final network state

NOTES

The action of a small Hopfield network for a two-sensor problem is shown here:

Operation is as follows:

Detections are received from each sensor – these determine the size of the Hopfield net.

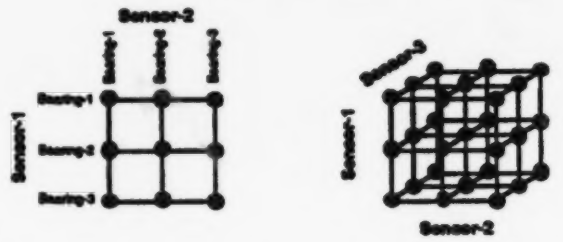
A probabilistic network is used to learn probability distributions for detections at each sensor. It also computes distance, i.e. how likely is it that a detection on sensor 1 represents the same target as a detection on sensor 2.

The distance matrix, together with auxiliary constraints determines the interconnections of the Hopfield net.

The Hopfield net is then initialized with small random values.

The network then iterates, reaching a stable state, which should indicate the most likely target locations.

Two Sensor and Three Sensor Hopfield Networks



Two Sensor Network

Three Sensor Network

NOTES

The approach used for the two-sensor problem requires a Hopfield network that is a two-dimensional array of neurons. The size of the array will, however, depend on the number of detections at each sensor.

For a three-sensor problem, a three-dimensional array of neurons is required – one dimension per sensor.

Further Work

- Problem of convergence to local minima
- Combining track initiation with track maintenance
- Feature selection at sensors
- Comparison with more traditional methods of constrained optimization

NOTES

Some directions for further work are indicated here.

One potential problem is the possibility of convergence of Hopfield networks to a local minimum. Some techniques that might be used to avoid this have been reported in the literature.

Another potential direction is to extend the neural network track initiation system to that of track maintenance. Extension of the track initiation system to multiple times as well as multiple sensors is one possible approach.

An important issue is feature selection. Obviously a Gaussian sum network that stores a complete Lofargram for each detection is not practical.

Another question is whether traditional techniques such as linear programming are in fact better and more reliable. Certainly, for the simple examples discussed in this talk a Hopfield net, or other optimizer, is hardly necessary – simple search will do.

NEURAL NETWORKS FOR INDEPENDENT RANGE AND DEPTH DISCRIMINATION IN PASSIVE ACOUSTIC LOCALIZATION

Pierre Zakarauskas, John M. Ozard and Peter Brouwer*

Defence Research Establishment Pacific, FMO Victoria, B.C. Canada, V0S 1B0.

***Datavision Computer Services Ltd. 203-1545 Pandora Ave., Victoria, B.C. Canada V8R 6R1.**

ABSTRACT

// Two feedforward neural networks with one hidden layer each were trained using a fast backpropagation algorithm to determine the position of an acoustic source in a waveguide. One network was trained to localize the source in depth while the other was trained independently to localize in range. The output layer consisted of one unit for each possible range or depth of the source. The networks were trained with a signal-to-noise ratio (S/N) of 50 dB and tested with patterns generated with S/N ranging from 0 dB to 20 dB. The performance of the neural networks (NNs) was compared with that of a nearest-neighbor classifier. Evaluation of the processors was done in the context of an estimation problem, i.e. by measuring the root-mean-squared (rms) error of the processors' estimates. The NNs turned out to be less resistant to noise than the conventional processor, but were faster. An explanation is given as to why multilayered feedforward neural networks cannot in general achieve the performances of optimum classifiers. //

CONTENT

PROBLEM STATEMENT

NN CONFIGURATION

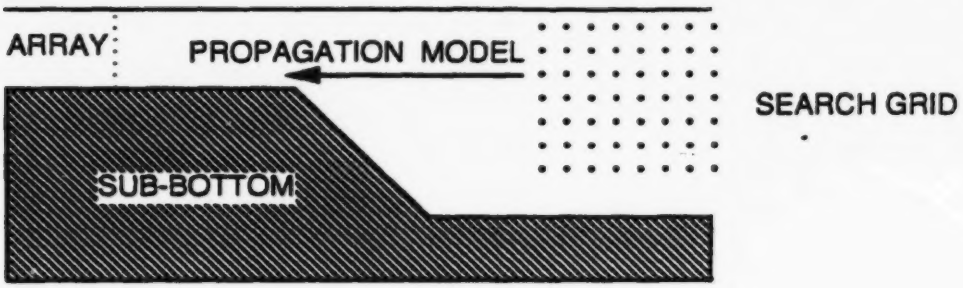
TRAINING ALGORITHM

POTENTIAL AND ACTUAL SPEEDUPS OVER CONVENTIONAL MFP

ROBUSTNESS ANALYSIS AND COMPARISON WITH MFP

INTERPRETATION OF RESULTS

MATCHED FIELD PROCESSING



COMPARE MEASURED FIELD AT ARRAY AND THE ONE PREDICTED BY A PROPAGATION MODEL

DISPLAY GOODNESS OF FIT - AMBIGUITY SURFACE

SEARCH AMBIGUITY SURFACE FOR BEST MATCH

MATCHED FIELD PROCESSING (CONT.) - IMPLEMENTATION DIFFICULTIES

1 - MFP NEEDS A RELIABLE AND ACCURATE MODEL OF SOUND PROPAGATION IN THE OCEAN. SHOULD INCLUDE:

- RANGE DEPENDENT PARAMETERS
- SHEAR IN THE SUB-BOTTOM
- ROUGHNESS SCATTERING
- EXPLICIT 3D

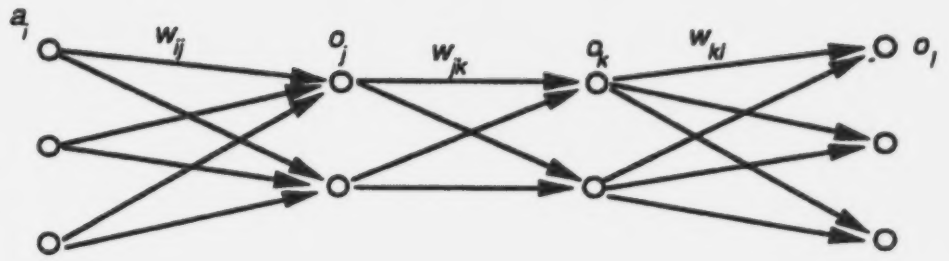
2 - PROPER MODELLING NEEDS DETAILED KNOWLEDGE OF ENVIRONMENTAL VARIABLES, SUCH AS:

- TOPOGRAPHY
- SOUND SPEED PROFILE
- SHEAR SPEED AND ABSORPTION
- ROUGHNESS

3 - MFP SEARCHES ALL POINT IN THE GRID. TIME OF SEARCH AND STORAGE REQUIREMENTS SCALES AS $d \cdot r \cdot a$

NEURAL NETWORKS

INPUTS HIDDEN LAYERS OUTPUT



$$o_j = \frac{1}{1+\exp\left[-\sum_i a_i w_{ij}\right]}$$

TRAINING:
- IMPOSE TARGET VALUES o_l FOR EACH SET OF INPUTS a

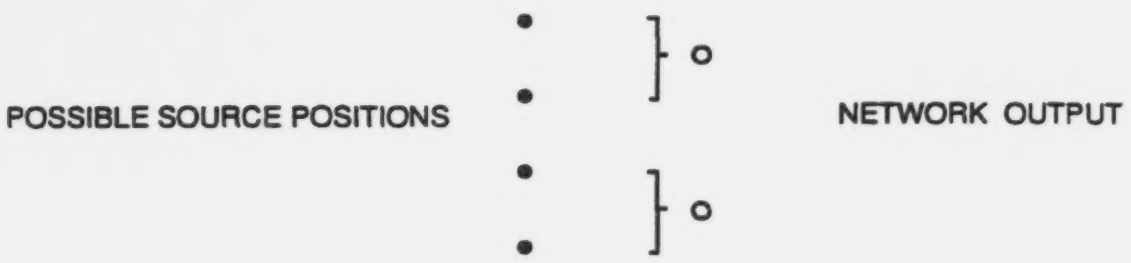
- ADJUST WEIGHTS THROUGH GRADIENT DESCENT (BACKPROPAGATION)

NEURAL NETWORKS (CONT.) - POTENTIAL ADVANTAGES

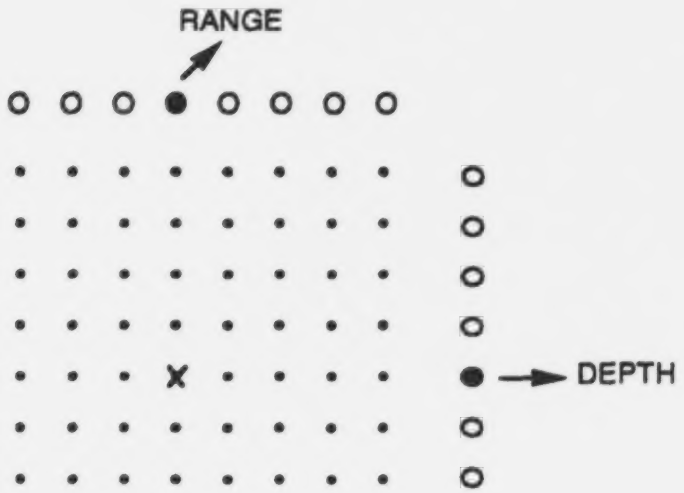
- CAN BE TRAINED ON REAL DATA

- CAN BE TRAINED TO GIVE DESIRED ANSWER

- GENERALIZE OVER SEVERAL DEPTHS OR RANGE



- PROCESS DEPTH AND RANGE SEPARATELY



NEURAL NETWORKS (CONT.) - POTENTIAL SPEEDUP OVER MFP

$$S = \frac{r \cdot d \cdot N_i}{N_i \cdot h + h \cdot N_o}$$

S : SPEEDUP FACTOR

r : NUMBER OF POSSIBLE SOURCE RANGES

d : NUMBER OF POSSIBLE SOURCE DEPTHS

N_i : NUMBER OF INPUTS

N_o : NUMBER OF OUTPUTS

h : NUMBER OF HIDDEN UNITS

TRAINING ALGORITHM

BACK-PROPAGATION

LEARNING RATE ε_i INDIVIDUALLY ADJUSTED FOR EACH WEIGHT

$$\varepsilon_i = \varepsilon_{i+1}\,\alpha_+ \quad \text{IF sign}(\Delta W_{i+1}) = \text{sign}(\Delta W_i)$$

$$\varepsilon_i = \varepsilon_{i+1}\,\alpha_- \quad \text{IF sign}(\Delta W_{i+1}) \neq \text{sign}(\Delta W_i)$$

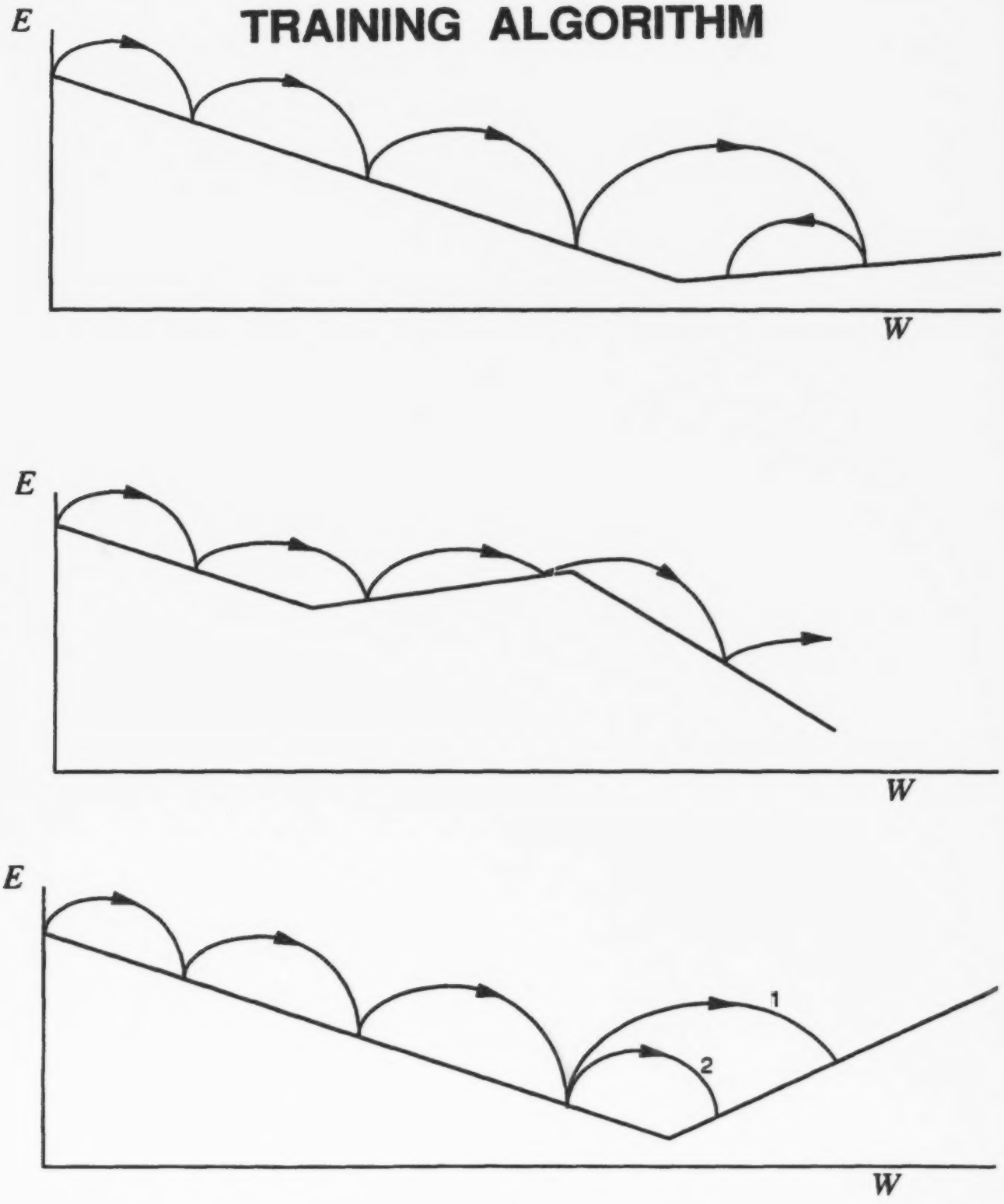
$\alpha_+ = 1.1$ $\qquad \varepsilon_{MAX} = 50$

$\alpha_- = 0.5$

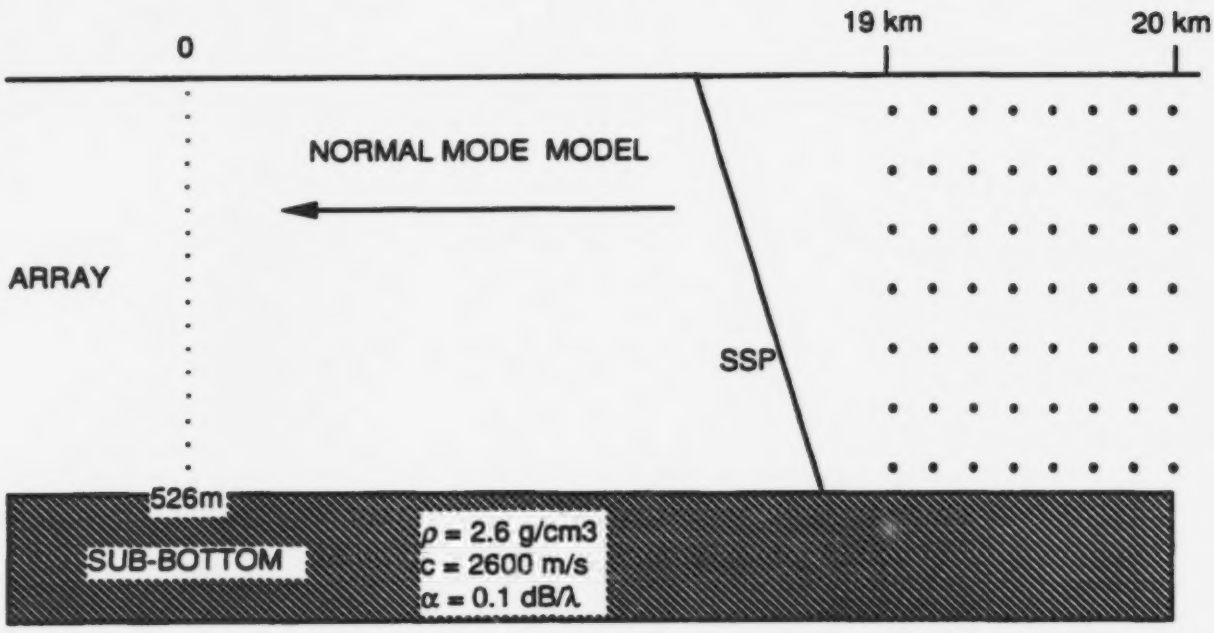
BACKTRACK IF $E_{i+1} > \beta\, E_i$ AND RETRY WITH

$\varepsilon_i = 0.5\,\varepsilon_{i+1}$

$\beta = 1.1$

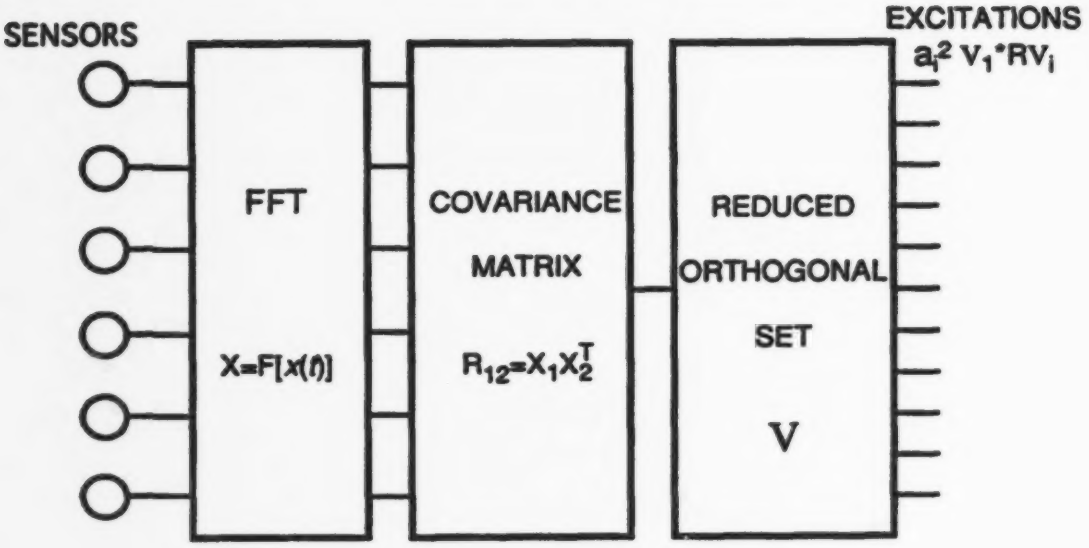

TRAINING ALGORITHM
E
W
E
W
E
1
2
W

ENVIRONMENTAL MODELLING



SEARCH GRID
22 depths X 11 ranges

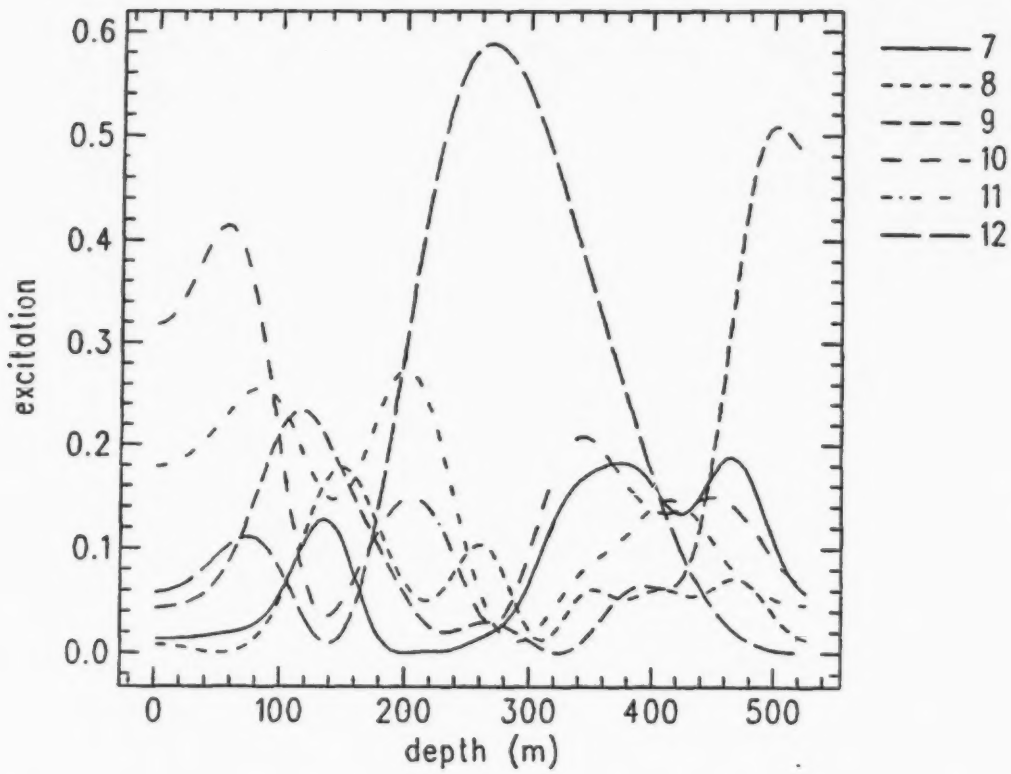
PREPROCESSING



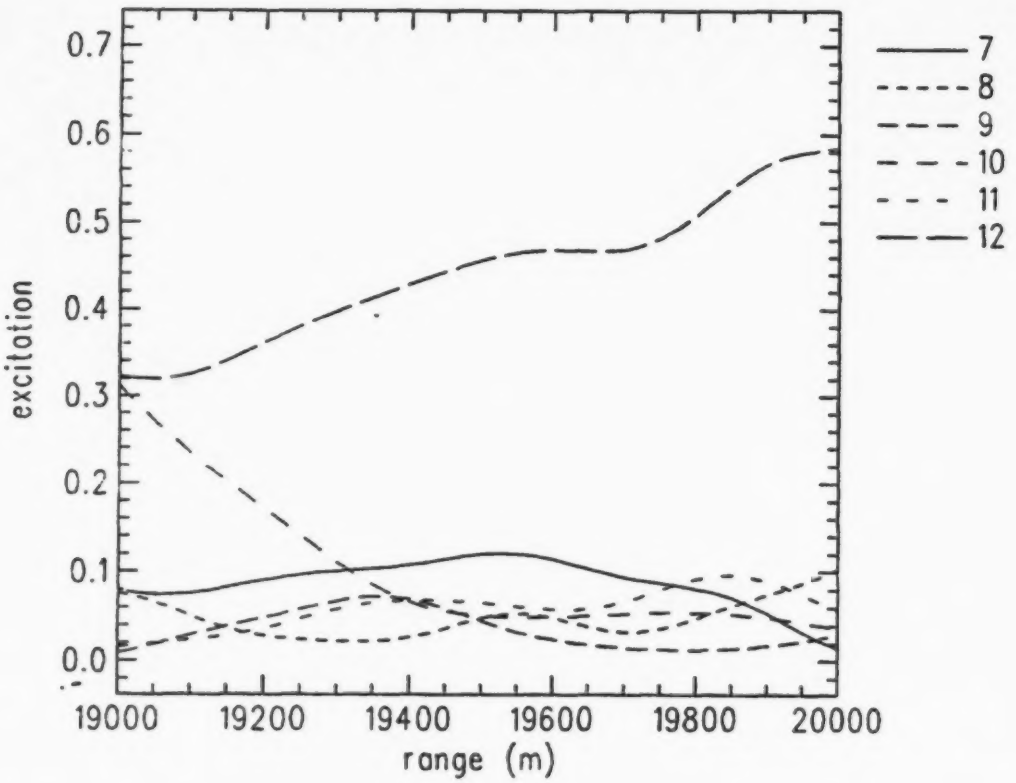
V: eigenfunctions of covariance matrix averaged over all source positions

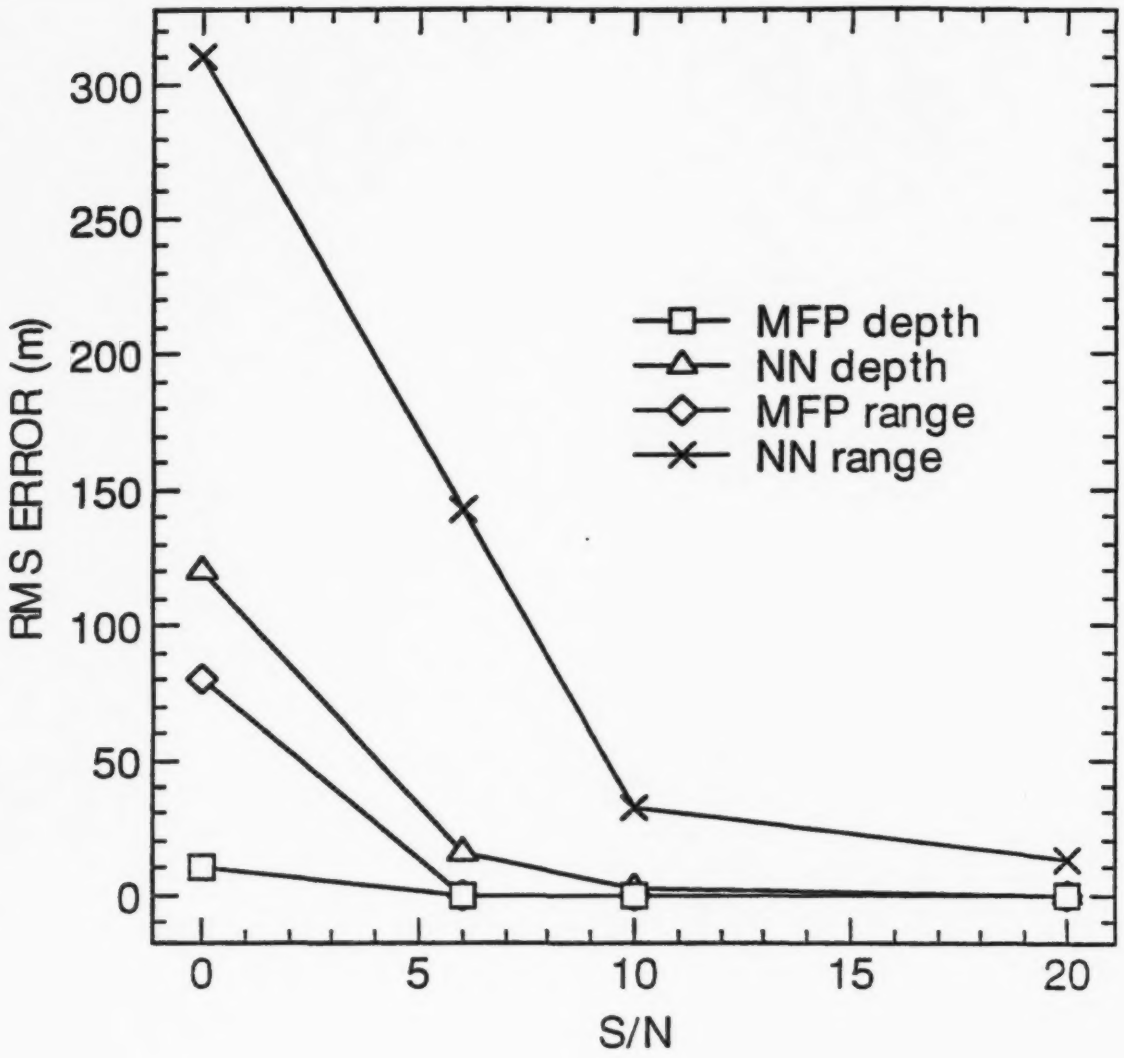
a: excitations of eigenfunctions V for R for a given source position

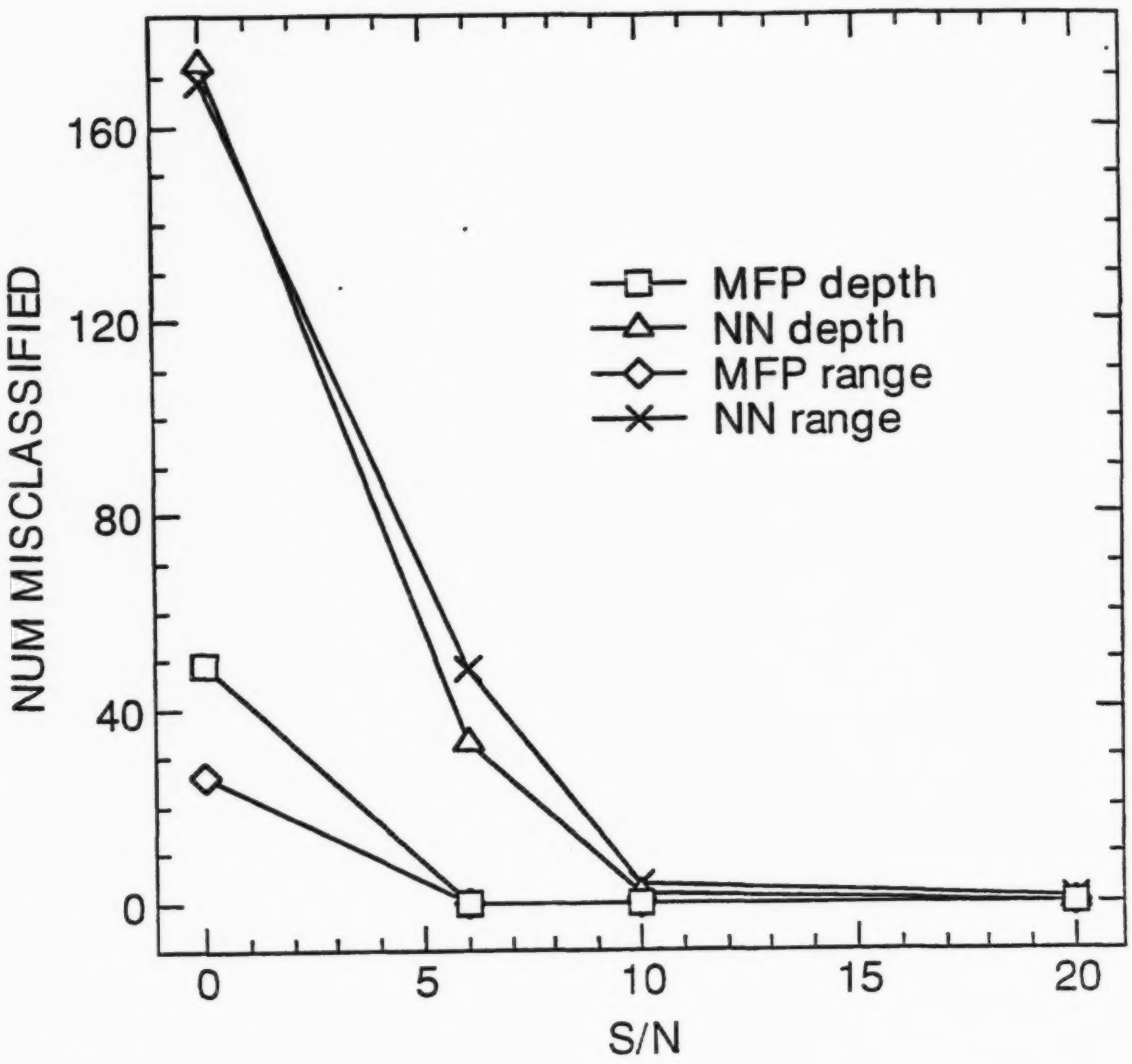
EXCITATION vs DEPTH FOR DIFFERENT EIGENVECTORS



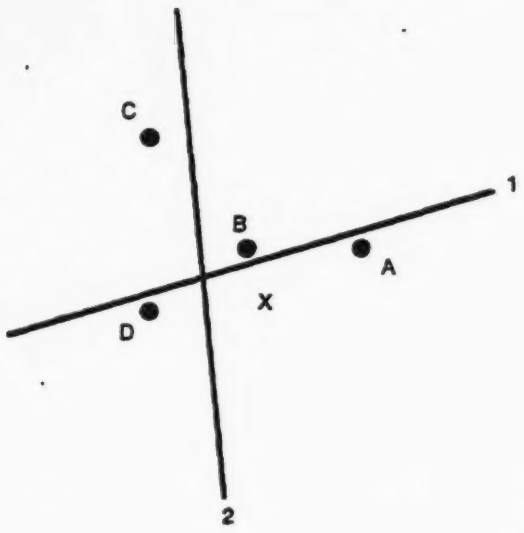
EXCITATION vs RANGE FOR DIFFERENT EIGENVECTORS



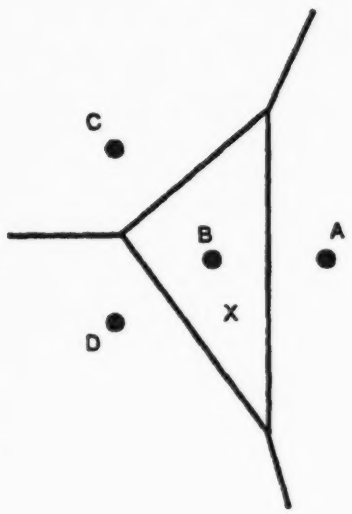

300
250
200
150
100
50
0
RMS ERROR (m)
0
5
10
15
20
S/N
MFP depth
NN depth
MFP range
NN range


NUM MISCLASSIFIED
160
120
80
40
0
0
5
10
15
20
S/N
MFP depth
NN depth
MFP range
NN range

INTERPRETATION OF NEURAL NETWORK DECISION SPACE



INTERPRETATION OF NEAREST-NEIGHBOUR DECISION SPACE



CONCLUSIONS

NNs WERE TRAINED TO LOCALIZE IN DEPTH AND RANGE

THE TRAINED NNs WERE TESTED WITH NOISE

- APPRECIABLE SPEED UP ANS STORAGE IMPROVEMENT
- NNs TO BE USED IN HIGH S/N

TRAINING DOES NOT ALWAYS SUCCEED, AND CAN BE VERY SLOW

NUMEROUS LOCAL MINIMA IN TRAINING ERROR SURFACE

Neural Networks Applied to the Phonemic Segmentation of Speech

by Roy C. Snell
Department of Mathematics
Royal Roads Military College

and

B.C. Dickson
Speech Technology Research Ltd.
Victoria, B.C.

Introduction

A technique that is frequently used to develop computer software for identifying and classifying patterns is the construction of an "Expert System". The design of such a classifier involves using human "experts" to determine a set of rules that will allow the system to carry out the desired discrimination tasks. Although rule-based expert systems may be implemented in traditional programming languages, advances in the field of artificial intelligence have provided a variety of tools specifically designed to ease the production of such systems. However, even with these tools, the selection, specification and testing of the rules to be applied by the system can be a difficult and time-consuming task.

Recent advances in the field of neural networks have provided an alternative approach to this classification problem. Using a corpus of data that has already been classified (by a human expert or some other manner) it is possible to design a neural network and train it to reproduce the classification results indicated on the training data. This approach completely removes the requirement for specifying the rules to be used and also greatly simplifies the final system implementation.

This paper will compare the classification results obtained from an "Expert System" based approach to the problem of identifying nasalized segments of human speech with some preliminary results obtained by training a Multi-Layer Perceptron to carry out the same task.

Background

The ability to segment the waveform obtained from human speech into a number of phonemic

classes is key to such areas of speech processing as speech recognition, speaker recognition, speaker verification and "word-spotting". An earlier study by the authors [1] addressed the problem of speaker verification. This task differs from that of speaker identification (sometimes referred to as "voiceprinting") in a number of key ways. Firstly, in this application we may assume that the speaker wishes to be identified correctly so we are dealing with a "co-operative subject" as opposed to a subject who is perhaps attempting to disguise his identity. Secondly, in the verification task we only require a yes/no decision as to whether the speaker is who he claims to be rather than a decision classifying the speaker as one of an open set universe of possible speakers. Finally, it is possible to gather a large set of training data for each subject from which the yes/no decision can be made. In fact, once the identity of an individual has been positively verified by other means, information from the speech data that was just processed during a verification sequence can be used to conveniently update the individual's database. The speaker verification problem is discussed in [2] and [3] and in Chapter 9 of Rabiner and Shafer [4].

The most significant challenge to making the correct decision in this task is the possibility that another individual will impersonate the correct subject by attempting to duplicate his voice quality and intonation patterns. The possible quality of such mimicry is evident when listening to professional entertainers who specialize in impersonating well-known individuals. Because of the likelihood that such attempts will be made to defeat the verification process, a system designer must avoid using speech parameters that can be duplicated by mimicry.

It was for this reason that our earlier study concentrated on the identification of "nasal murmurs" in the speech waveform so that the verification parameters would be extracted during the production of nasalized speech. These events occur during voiced speech when the majority of the pulmonary energy is deflected through the nasal cavity. The resonant behaviour of the speech that is produced during such an event is determined chiefly by the geometric configuration of the individual's nasal cavity and is not, therefore, susceptible to the duplication by an impersonator.

Preliminary Waveform Processing

The waveform analysis used in the original study and for this paper involves 10th order, fixed frame, Linear Prediction (LPC) (see Markel and Gray [5] for further details of the method) . The input speech is sampled at 10 KHz with a 12 bit resolution and the resulting digitized samples are segmented into 200 point frames (each representing a 20 millisecond segment of the original speech). The data from each frame is pre-emphasized and a Hamming window is applied. The resulting data is then processed using the standard autocorrelation method for linear prediction to produce 10 reflection coefficients (RC's) . Additional parameters extracted at this time are the normalized residual energy (RENG) derived from the LPC analysis and the number of zero crossings (0x) in the frame . Of the information obtained in this manner from the speech waveform, the RC's are known to extract resonance characteristics of the vocal tract that relate to the articulatory settings of the speaker for the phone being produced; the RENG relates the

excitation energy to the total speech energy during the frame; and the 0x value is useful in distinguishing voiced speech from unvoiced speech or silence,

From the RC's, a number of alternative representations of the vocal tract response during the frame may be extracted. These include the filter coefficients (A's) of the all pole filter

$$A(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{10} z^{-10}}$$

that is determined by the RC's; the cepstral coefficients (CEP's) derived from the A's; and the frequencies and 3db bandwidths of the formants (resonances) in the vocal tract response (FBW's). All of these additional parameters may be derived from the RC's (see Chapter 10 of [5] and the discussion in [6]).

The LPC based model of speech production is the basis for the majority of work in the areas of speech analysis and synthesis. In particular, the success of the model for speech synthesis indicates that it effectively captures information concerning the response of the vocal tract during articulation. Results from our previous study indicate, as well, that the model preserves speaker-dependent characteristics.

Expert System Design and Performance

In developing the Expert System for identifying nasal frames, a corpus of test data was recorded consisting of 880 short sentences from 10 male speakers. The total duration of the test data was approximately 1100 seconds of speech (54,000 frames). The individual utterences were manually classified into nasal and non-nasal frames by an expert phonetician (the second author) using a combination of aural and waveform information. This resulted in 5,700 frames that were classified as nasal murmurs. The data was then processed as described in the previous section so that all of the parameters, including RC's, A's, CEP's and FBW's were available for each frame.

The "expert" then began to examine the relationship between these parameters and nasal murmurs, guided by behaviour that was expected from a variety of theories of speech production. After extensive investigation, a set of rules was specified that could be used to classify the frames containing nasal murmurs. These rules used information from the following parameters : 0x, RENG, filter coefficients (specifically a_2) and the location of the first 3 formant frequencies.

When the Expert System incorporating these rules was applied to the test corpus, 7000 frames were identified as nasalized. The confusion matrix for the results is :

		Actual Type: Nasal	Actual Type: Non-Nasal	
Identified Type	Nasal	5400	1600	7000
	Non-Nasal	300	48300	48600
	Total	5700	49900	

Table 1 - Confusion Matrix for 1983 Study

Since processing in the next stage of the speaker verification system assumes that only nasal frames have been extracted, the key performance parameter of the expert system is that 77.1% (5400 / 7000) of the nasal frames identified were, in fact, nasal murmurs. The overall system will never be faced with a situation in which too few nasal frames will be present in the test utterence since we are dealing with a "co-operative" user who wishes to be identified. This allows the system to specify the exact test phrases to be used and they may be selected to contain a sufficient amount of nasal murmur activity. For example, one test phrase used in the data base was

" My number is BCM-199-379 D "

which contains a high number of nasal frames.

Since this paper is not specifically concerned with the performance of the verification portion of the system, we will not discuss that topic other than to note that, for each of the subjects, features were extracted from all of the nasal frames in the test data (as identified by the expert system). Using these features, a series of 2200 tests was carried out using sets of 30 nasal frames extracted (again by the expert system) from utterences recorded from the subjects between 3 days and 1 week later than the original recordings. The overall average of correct subject identification was 97.86% (including no errors for 7 of the subjects and most of the error due to an error rate of 16.5% for one of the subjects)

Neural Network Design and Performance

The development of Neural Networks (NN's) as a tool for pattern classification suggests an alternative method to the "Expert System" approach to the identification of nasal frames. In fact, neural networks might also be expected to perform well in the second stage of the verification task but we will not discuss that application in this paper.

To examine the effectiveness of the neural network approach, a limited amount of test data was captured from four subjects (two male and two female); analyzed according to the procedures outlined above; and manually segmented into nasal frames.

Using guidelines derived during the development of the expert system concerning the importance of the available parameters to the classification task, the following pattern vectors were chosen to represent each frame :

Input Parameters : (10)
- Zero crossing count (0x)
- Filter coefficients (a_2 and a_3)
- Normalized energy
- Formant frequencies (F_1 , F_2 and F_3)
- Formant Bandwidths (B_1 , B_2 and B_3)

The initial NN experiments used a single output node taking on values of 0 for non-nasal frames and 1 for nasals. In carrying out the manual classification of the speech data, a human expert establishes the boundaries of each nasal segment. This can produce a problem, however, when the locations of the individual 20 millisecond analysis frames are superimposed on the waveform for classification. Some frames contain no nasal activity (a score of 0); others lie completely within a nasal segment (a score of 1); while the remaining frames ocurring at the onset or termination of nasalization will contain portions of both nasal and non-nasal activity.

During the first investigations, a number of thresholds (with values between 25% and 75%) were selected and a frame was classified as nasal if the fraction of nasal activity in the frame exceeded the chosen threshold. Several NN architectures were trained with data classified according to this method and the trained networks were used to classify additional test data. An examination of the classification errors that ocurred on both the training sets and the test data indicated that the contradictory behaviour of the parameters in transitional frames might be seriously affecting the network performance. The fact that most nasal murmur activity is only a few frames long means that the number of transitional frames in an utterence is not inconsequential.

Since node output values in the range 0 to 1.0 are standard for a NN, the second set of experiments used the scores representing the fraction of nasalization in the frame directly. It was expected that the more gradual transitions between 0 and 1 in the data would enhance the ability of the network to adapt to the transitions in the input parameters. Networks that were trained on this new data format performed significantly better that those using the hard-limited output values described above. The results seemed to indicate the usefulness of the third type of "mixed" frame.

The existence of three classification types (non-nasal, mixed and nasal) suggested an alternate NN toplogy where a seperate output node is assigned to each type and the goal of the network is to produce an output value of 1 on the correct node for each input training vector. In this format the correct output corresponding to each training vector is (1,0,0) for a non-nasal frame, (0,1,0) for a partially nasal frame and (0,0,1) when the frame is completely nasalized. Actual network

outputs might appear as (0.002, 0.9678, 0.0832) which clearly suggests a transitional frame; (0.1023, 0.0890, 0.9156) which suggests a nasal frame; or (0.3425, 0.5432, 0.2314) which may be interpreted as confusion on the part of the network as to the actual classification type. The additional information available in this format can prove very useful in improving the decision-making process.

Preliminary experiments using the three output nodes established clearly superior performance. As a result, the neural network used for the remainder of our investigation was a multi-layer perceptron with the following topology :

Input Layer :	10 nodes corresponding to the 10 input parameters connected to the output nodes and all nodes in the hidden layer
Hidden Layer :	10 nodes fully connected to each input node and to the output nodes
Output Layer :	three nodes corresponding to non-nasal, mixed and nasal frames

The network therefore contains 23 nodes and a total (including thresholds) of 173 weights.

In the process of adapting the classification problem to the neural network format, a number of advantages to the NN approach become apparent. First, the ability to classify a frame as being partially nasal is simple to implement in the NN structure but very difficult (if not impossible) in a rule based system. In addition, while it was known that formant bandwidths are significant in the nasal classification process, the complexity of the formant relationships was too great for the human expert to incorporate this information in the selection rules. It is, however, easily added to the network model. Finally, while the rule-based system was required to provide a set of selection criteria to be used for all speakers, the NN approach permits the training of separate networks to the nasal characteristics of each individual speaker.

Since the training of a single network to classify nasals for all speakers is the most difficult task, the initial investigations with the network described above examined the classification problem for individual speakers. In order to make the required classification of a frame, two thresholds were set at 25% and 75%. A frame containing less than 25% of nasal activity is classified as non-nasal; more than 75% of nasal activity results in a nasal classification; and all other frames are classified as mixed.

Once a decision had been made relating to the network architecture, the data from the two male and two female female subjects was processed in a number of ways. Each experiment involved the identification of a desired test group; the random selection of frames from that test group to be used in training the network; and finally the application of the trained network to the remaining frames in the group to establish recognition accuracy.

In total, seven different groups were selected and the same tests were carried out on each group. In order to determine classification accuracy for networks trained to recognize the nasals for a single speaker, four of the groups involved only data from the individual speakers; to examine performance based on similarities in voice quality, the two male and two female speakers were grouped together; and the final group consisted of the data from all four speakers. Table 2 indicates the number and type of frames data available for each of the seven groups tested.

Table 2

Group #	Subjects	Nasal	Mixed	Non-Nasal	Total
1	Male 1	129	39	456	624
2	Male 2	215	52	682	949
3	Female 1	150	54	559	763
4	Female 2	285	47	947	1279
5	Males 1 & 2	344	91	1138	1573
6	Females 1 & 2	435	101	1506	2042
7	All Subjects	779	192	2644	3615

For each of these seven groups, a network was trained using a randomly selected training set. The trained network was then used to classify the remainder of the frames available for the group. Tables 3 to 11 present the classification results for the seven neural networks and for the application of the Expert System developed in the 1983 study to the test data. The size and composition of the randomly selected set of training data is indicated below each table. In addition to the confusion matrix describing the classification results, two parameters are reported beside each table . The parameters are :

Accuracy - defined as the % of frames classified as nasal that are actually nasal. The accuracy is reported as two values with the first calculated on the basis that an error has ocurred when a mixed frame is classified as nasal while the second accepts mixed frames as nasal.

Capture Ratio - the % of nasal frames present in the data that are classified as nasals by the system. Again, two values are reported in which mixed frames are distinguished from nasals and in which the two classes are combined.

From the point of view of the speaker verification task that is to be carried out following the nasal frame classification, it is more important that the value of the **Accuracy** parameter be as high as possible. Even if the **Capture Ratio** is low, we expect the input speech to contain a sufficiently large number of nasal frames to permit the verification stage to proceed.

With these comments in mind, we note a number of significant features that emerged as the tests were carried out.

Expert System Performance on the Data Set

Table 3 presents the results obtained when the Expert System developed in the 1983 study was used to classify the nasal frames in the four speaker data set. In the earlier study, no special consideration was given to frames that were only nasalized for a portion of their duration. The first confusion matrix in Table 3 indicates the performance of the Expert System when partially nasal frames are considered to be nasal. In comparing the results with the overall performance reported in Table 1 for the original study, both the accuracy (62.6% compared to 77.1%) and the capture ratio (67.1% compared to 94.7%) are seen to be a good deal lower.The data in the second matrix represents the results as they relate to the added class of mixed frames that has been added for this study. Note that the numbers of frames in each class as predicted by the Expert System is unchanged, however the accuracy has decreased now that an error is assumed if a mixed frame is classified as nasal. The corresponding increase in the capture ratio is the expected behaviour when the accuracy parameter decreases in value.

Threshold Comparisons for Neural Net Classification

Once a neural net has been trained to the desired performance level on the training data, the trained network is then used to classify the remaining data in the test group. The desired output for each frame is a vector containing a single value of 1 (at the output node corresponding to the correct class of the frame) and 0 values at the other two nodes. In fact, the true output will only approximate the desired behaviour. It is therefore necessary to choose a threshold value (between 0 and 1) and treat node outputs that are above this threshold as 1's and those below the threshold as 0's. The choice of threshold significantly affects the classification performance of the network.

The data in Table 4 illustrates how varying the threshold from 0.5 to 0.9 affected the performance of one of the networks tested. As the threshold was raised the accuracy increased from 76% to 85% with a corresponding decrease in the capture ratio of 55% to 40%. The behaviour exhibited in this case was duplicated in all of the other tests. The ability to control network performance by adjusting the threshold is a very desirable feature of the neural net approach which would be difficult, if not impossible, to duplicate in an expert system. In order to approximate a 50% capture rate, a threshold of 0.75 was used in all of the neural network performance evaluations.

One danger in the neural network approach to data classification is "overtraining" the network. The use of optimization techniques guarantees that each successive pass through the training data will decrease the classification error. If training continues too long, the possibility exists that the

network will be so "in-bred" to the training set that it will not be able to recognize the desired patterns on other test data. The information in Table 5 was obtained by continuing to train the network from Table 4 for twice as many training passes. The change in accuracy when this network was used to classify the remainder of the data was nominal (an increase from 82.3% to 82.9%) and clearly indicates that overtraining is not a problem for this application.

Neural Network Performance on the Data Sets

Results for the seven test groups described in Table 2 are presented in Tables 6 to 11 and the 0.75 threshold data in Table 3. The classification performances were quite uniform with accuracy values above 80% (with the exception of Male Subject 1) and capture ratios near 50% (except for Male Subject 2 where an 86% accuracy was combined with only a 36% capture ratio). The following points are significant :

- the accuracy of the neural network that was trained on data from all four subjects was significantly higher than that of the expert system. If we distinguish mixed frames from nasals, the figures were 82.3% compared to 50.2% while combining the two groups results in accuracies of 88.6% and 59.5% respectively. This represents strong evidence in support of the neural network approach to nasal classification.

- for most of the tests, the training data was randomly selected from all of the available data for the group resulting in a distribution of frame types in the training set consistent with the distribution for the entire group. In order to investigate the effects of a different distribution of training data, the random selection for Female Subject 2 (Table 9) was carried out to balance the number of frames of each type in the training set (34 mixed / 34 nasal / 36 non-nasal). The 87% classification accuracy for the resulting network was consistent with the results obtained in the other cases and suggests that the neural network performance is not significantly affected by the distribution of frame classes in the training set.

- although not apparent from the performance data presented, the prediction error decreased rapidly during training of all the networks so that a relatively small number of training passes was required. In addition, as previously mentioned, none of the networks exhibited performance reduction due to "overtraining". These two facts indicate that the three classes used are well defined and that the data used to carry out the classification task is able to accuractely distinguish them.

Summary

Although testing is required on a much larger data set, preliminary indications suggest that the neural network approach will prove to be an effective tool for classifying nasal murmurs in

human speech. The use of three output nodes corresponding to nasal, non-nasal and mixed frames appears to significantly improve the classification efficiency. Since separate networks can be trained to recognise the nasal characteristics of individuals or groups, the process can be easily adapted to any application. In addition, the ability to increase the classification reliability by adjusting the value of the detection threshold is very useful.

Compared to an expert system in which the derivation of the rules can be difficult and time-consuming, the use of a neural network permits easy inclusion of any number of the speech parameters with the correct relationships between the parameters and the event classifications determined during the training process.

References :

1. B.C. Dickson and R.C. Snell, "An Investigation of a Speaker Verification System using Parameters derived from the CRC Vocoder," Centre for Speech Technology Research Technical Report SV8301, 1983

2. A.E. Rosenberg and M.R. Sambur, "New Techniques for Automatic Speaker Verification," IEEE Trans Acoustics, Speech and Signal Processing, Vol ASSP-23, No. 1, pp67-72, February 1975

3. B.S. Atal, "Automatic Recognition of Speakers from their Voices," Proc IEEE, Vol 64 , No. 4, pp 460-475, April 1976

4. Lawrence R. Rabiner and Ronald W. Schafer, "Digital Processing of Speech Signals," Prentice-Hall, 1978

5. J.D. Markel and A.H. Gray Jr., "Linear Prediction of Speech," Springer-Verlag, 1976

6. R.C. Snell and F.Milinazzo, "Formant Extraction from LPC Analysis Data," IEEE Transactions on Signal Processing, April 1993 (to appear)

Expert System Performance
Four Speaker Data Set

Actual

Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	-	715	428	1143
Non-Nasal	-	352	2120	2472
Total	-	1067	2548	3615

Accuracy = 62.6%

Capture Ratio = 67.1%

Omitting Mixed Classification

Actual

Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	106	574	463	1143
Non-Nasal	86	205	2181	2472
Total	192	779	2644	3615

Accuracy = 50.2% (59.5%)

Capture Ratio = 73.7% (70.0%)

Including Mixed Classification

Table 3

Network Trained on all Four Subjects
Threshold Comparisons

Threshold = 0.5

Actual

Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	51	406	74	531
Non-Nasal	134	333	2437	2904
Total	185	739	2511	3435

Accuracy = 76.4% (86.1%)

Capture Ratio = 54.9% (49.4%)

Threshold = 0.75

Actual

Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	27	354	49	430
Non-Nasal	158	385	2462	3005
Total	185	739	2511	3435

Accuracy = 82.3% (88.6%)

Capture Ratio = 47.9% (41.2%)

Threshold = 0.9

Actual

Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	18	292	34	344
Non-Nasal	167	447	2477	3091
Total	185	739	2511	3435

Accuracy = 84.9% (90.1%)

Capture Ratio = 39.5% (33.5%)

Training Set : 180 frames (7 mixed / 40 nasal / 133 non-nasal)

Table 4

Network Trained on all Four Subjects
Additional Training Passes

Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	27	345	44	416
Non-Nasal	158	394	2467	3019
Total	185	739	2511	3435

Accuracy = 82.9% (89.4%)
Capture Ratio = 46.7% (40.3%)

Table 5

Network Trained on Male Subject 1
Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	4	55	19	78
Non-Nasal	28	59	377	464
Total	32	114	396	542

Accuracy = 70.5% (75.6%)
Capture Ratio = 48.2% (40.4%)

Training Set : 82 frames (7 mixed / 15 nasal / 60 non-nasal)

Table 6

Network Trained on Male Subject 2
Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	5	69	6	80
Non-Nasal	37	124	588	749
Total	42	193	594	829

Accuracy = 86.3% (92.5%)
Capture Ratio = 35.8% (31.5%)

Training Set : 120 frames (10 mixed / 22 nasal / 88 non-nasal)

Table 7

Network Trained on Female Subject 1
Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	9	86	3	98
Non-Nasal	37	47	504	588
Total	46	133	507	686

Accuracy = 87.8% (96.9%)
Capture Ratio = 64.7% (53.1%)

Training Set : 77 frames (8 mixed / 17 nasal / 53 non-nasal)

Table 8

Network Trained on Female Subject 2
Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	2	134	18	154
Non-Nasal	11	117	893	1021
Total	13	251	911	1175

Accuracy = 87.0% (88.3%)
Capture Ratio = 53.4% (51.5%)

Training Set : 104 frames (34 mixed / 34 nasal / 36 non-nasal)

Table 9

Network Trained on Male Subjects 1 & 2
Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	18	188	24	230
Non-Nasal	61	125	1005	1191
Total	79	313	1029	1421

Accuracy = 81.7% (89.6%)
Capture Ratio = 60.1% (52.6%)

Training Set : 152 frames (12 mixed / 31 nasal / 109 non-nasal)

Table 10

Network Trained on Female Subjects 1 & 2
Threshold = 0.75

	Actual			
Predicted	Mixed	Nasal	Non-Nasal	Total
Nasal	10	247	18	275
Non-Nasal	83	166	1388	1636
Total	93	413	1406	1912

Accuracy = 89.8% (93.5%)
Capture Ratio = 59.8% (50.8%)

Training Set : 130 frames (8 mixed / 22 nasal / 100 non-nasal)

Table 11

#516134 # 5

CRAD SIPWG Workshop on Neural Networks
DREA, 6-7 May 1992

AUTOMATIC MODULATION RECOGNITION

René Lamontagne
Communications EW Section
Electronic Warfare Division
Defence Research Establishment Ottawa

1.0 Background

From the earliest days of radio communication, the need to monitor the electromagnetic signal activity in the Radio Frequency (RF) spectrum has existed. Civilian authorities have always wished to monitor the transmissions over their territory to maintain a control over this activity. Similarly, military organizations have tried to monitor the radio activities of other powers for reasons of national security, gathering intelligence information in peace time, and assessing strategic and tactic planning during war time in light of information obtained from intercepted signals.

Typically, the technique employed by monitoring stations is based upon a well-tried technique that has been used for many years [1]. Using a continuously tunable general purpose receiver, the operator searches through the RF spectrum hoping to make an interesting interception. One variation to the scenario, is the one man-two receivers situation: one is used for searching while the other one remains tuned to a known interesting frequency. A further variation, called the master-slave technique, involves a group of operators. The master operates a fast-tuning sweep receiver with an associated panoramic display unit. When a signal of possible interest is located, the signal center frequency is transferred to one of the slaves, who tunes to the appropriate frequency and does the monitoring with a conventional general purpose continuously tunable receiver. Neither of these techniques, however, overcomes the fundamental problem of serious overcrowding of the radio spectrum. An operator searching through a multitude of noisy signals, of which most are uninteresting, is confronting a very boring and time consuming task.

To address the problem, Land Forces D-Sigs-EM has initiated a task called SCAR (Signal Classification And Recognition) for the development of an intercept/analysis test-bed by CEW (Communications EW Section). Also, an in-house project called ACES (Advanced Communication Electronic-Support-Measures System) started in 1991 [2]. Using state-of-the-art technology to process a huge amount of data with sophisticated algorithms, the system will, hopefully, be able to assist and to increase the efficiency of an operator in monitoring the High Frequency (HF) and Very High Frequency (VHF) bands.

The ACES test-bed, in its preliminary stage, will consist of a wideband digital receiver, an energy detection subsystem (Fast Fourier Tranform (FFT) processors and wideband Digital Signal Processors (DSPs)), digital drop receivers (DDR), and a narrowband DSP subsystem (see Table 1 and Viewgraph 3). At a further stage, more receivers will be incorporated as well as a sophisticated DFACTT (Data Fusion And Correlation Techniques Test-bed) subsystem developed by CEW, which has already successfully passed through field trials with the Land Forces. This subsystem will provide to the ACES test-bed, very high level man-machine-interface abilities, increasing significantly the operator's efficiency in interpreting intelligence information.

As part of the narrowband subsystem, Automatic Modulation Recognition (AMR) algorithms are currently being investigated. Assuming that a signal has been detected, isolated, and down-converted, the task of the AMR device is to identify the signal modulation type among several ones known to be in use in a given environment. Such devices are already commercially available but it appears, according to the

Table 1: Principal components of the ACES test-bed.

ACES		
Digital Receiver	2 MHz Instant Bandwidth, 85 dB Dynamic Range, HF/VHF bands	Steinbrecker (USA)
FFT processors	65536 points, 10 ms	Austek (Australia)
wideband DSPs	SUN VME, 160 MFlops, multiprocessors (8 cells), linearly expandable	Intel's iWarp system (USA)
Digital Drop Receivers (DDR)	VME programmable, Dynamic range up to 96 dB, Bandwidth programmable from $F_o \div 2^{15}$ to $F_o \div 2$	Contracted to CAL (Ottawa), design based on Harris DDF and NCOM chips
narrowband DSPs	SUN VME, TMS320C30 or C40 multiprocessors	Contracted to ASIC (Ottawa)

specifications [3] [4], to provide insufficient accuracy for ESM applications, where low Signal-to-Noise Ratio (SNR) signals are common. Therefore, the intent is to develop a more robust algorithm to support ACES and SCAR projects.

Since the AMR activities have started, three sub-projects have been initiated. First, in-house work is currently being carried out on time-domain features, of which results will be presented in Section 2.0. Also, frequency-domain features are being investigated under contract. This topic will be covered in Section 3.0. Finally, CEW is involved as scientific advisor for a Defence Industrial Research (DIR) program on genetically enhanced neural networks. As part of this project[1], the firm International Neural Machines (INM) Inc. will use the expertise developed under the previous projects to investigate the potential of the genetic algorithm to enhance the performance of the modulation recognition engine. Finally, hardware implementation will commence after completion of the algorithm.

[1] The three major parts are:
- a. character recognition for INM
- b. sonar signal identification for DCIEM (Defence and Civil Institute of Environmental Medicine)
- c. AMR for DREO

2.0 Time-Domain Features - PRELIMINARY RESULTS

2.1 The features

Since modulation is accomplished by using a baseband signal to alter the amplitude or phase of a carrier, using time-domain features has always been the classic way to perform AMR [6]. However, the results obtained, although good for high SNR, were not good enough for ESM applications. The intent here is to investigate the use of a more exhaustive set of features in combination with an Artificial Neural Network (ANN), which has proven its ability to map complex decision regions and has an inherent capability to perform fast classification, when appropriate hardware is used.

A modulated signal can be expressed by

$$s(t) = A(t)\cos(\omega_c t + \varphi(t) + \theta_0), \tag{1}$$

where A(t) and φ(t) are baseband signals representing the information message to be transmitted, ω_c is the carrier frequency, and θ_0 is the initial phase angle.

An alternative signal representation, more practical for DSP purposes, is called the analytic signal and is denoted z(t). It is a complex bandpass signal with a one-sided spectral density Z(f), and is obtained by adding the Hilbert transform of s(t) as the imaginary part, i.e.,

$$z(t) = s(t) + j\hat{s}(t) \tag{2}$$

and

$$Z(f) = \begin{bmatrix} 2S(f), & f>0 \\ 0, & f<0 \end{bmatrix} \tag{3}$$

Often, the analytic signal is expressed in terms of In-phase and Quadrature components, denoted $S_I(t)$ and $S_Q(t)$, corresponding respectively to s(t) and its Hilbert transform. Using that signal representation, one can estimate the signal envelope and instantaneous frequency [5], which appears to provide sufficient information to discriminate among the modulation types as we will show in the following sections. The instantaneous frequency is used rather than the phase to avoid having to estimate the carrier frequency.

Using the signal envelope and instantaneous frequency as parameters, three other parameters are calculed for a total of five time-varying parameters: the signal envelope, A(t); the instantaneous frequency, F(t); the squared envelope, $A^2(t)$; the envelope multiplied by its derivative, $A(t)\cdot A'(t)$; and the squared envelope multiplied by the instantaneous frequency, $A^2(t)\cdot F(t)$.

As detailed in [6], diverse combinations of those parameters have been used by various researchers to estimate time-invariant features useful for signal classification. In our case, the feature extraction process (see Viewgraph 4) consists of taking a sample of about 8192 points from the complex analytic signal in order to compute the five parameters. Then, some statistics are calculated to create the following set of 21 time-invariant features:

(1-15) three statistical moments (the standard variation, the skewness, and the kurtosis) are calculated for each parameter
(16) the ratio of the variance of A^2 over the squared mean of A^2
(17) the variance of the variance of A^2
(18) the variance of the short-term energy (20 ms) normalized by the mean energy
(19) the difference between the maximal and minimal values of the short-term energy normalized by the mean energy

(20) the probability that A^2 be smaller than 5% of the mean value
(21) the maximum value of A divided by the average of A

2.2 The classification process

Typically, for AMR, probabilistic linear classifiers have been used. They are fast and easy to implement [7]. However, they are limited in many ways and offer disappointing performance as the problem dimensionality and complexity increase. Since, here the complexity of the problem has been significantly increased by adding features and diminishing the SNR, the use of a more robust classifier is proposed, such as a multilayer ANN. Using a more sophisticated classifier and a bigger size pattern vector, we hope to achieve better performance and successfully classify signals having low SNR's.

A k-Nearest-Neighbour (kNN) classifier has been used to validate the training and testing data sets [8]. Neither the linear classifier nor the ANN would be really appropriate for that purpose. However, the intent is not to use the kNN for AMR, being too slow and requiring too much memory. Rather, the intent is to use a multilayer Backpropagation (BP) hoping to achieve as good as or better than the kNN classifier.

For the BP, three architectures have been investigated: no-hidden layer, one-hidden layer, and two-hidden layers (See Viewgraph 5). The first architecture was used to establish a comparison with the probabilistic linear classifier, since they both have the same number of weights. However, one uses a statistical approach to adjust the weights, while BP uses "learning", an iterative process.

Contrary to the previous case, which was limited to linearly separable problems, the second architecture, using one hidden layer, is suitable for arbitrarily complex decision regions, convex or not, as shown in [WIELAND] and [HUANG]. The main question remains to determine the optimal number of hidden units. Since this number is data set dependant, an attempt was made to find it empirically. The number of hidden units was set to ½, 1, and 2 times the number of input nodes. The layers are fully inter-connected.

Finally, a two-hidden layer BP was trained on the data set, to investigate a possible advantage over the one-hidden layer ANN. Based upon intuitive rules described in [9], [10], and [11], there should be one node per convex region (hypercube) per class in the second hidden layer (see Viewgraph 6). Assuming 3 such regions, 15 nodes were used. For the first-hidden layer, the problem is more subtle. According to the same references, one can deduce that 630 units would be required (2 times 21 features times 15 hupercubes in the second hidden layer). Note that the first and second hidden layers would be locally rather than fully connected. However, assuming that most hyperplanes are shared by adjacent classes and that the layers are fully connected, this number was reduced to 45 nodes in the first-hidden layer, which number is in accordance with the ratio 3:1 recommanded by Kudrycki in [12], and appeared to be a fairly good choice also in [13], page 243. To find the best combination, more experimentation would be required, however this configuration will be suitable to decide whether or not two hidden layers are advantageous for this particular problem.

For the simulation of the ANNs, the Xerion software was used on a SUN IPX workstation. Xerion is a collection of C libraries, including routines for contructing networks, displaying them, and training/testing them, that were built by the Department of Computer Science, University of Toronto. The BP paradigm, which has been implemented using those libraries, was used for the simulations.

The networks were initialized with random weights on the range -0.1 to 0.1. Small values were chosen to avoid premature saturation [14]. For learning, the momentum descent algorithm was used. The step size was set to 0.1 and the momentum to 0.9. Keeping those values constant, the network was trained using online learning until convergence was reached. To get closer to the optimal value (minimal error), rather than changing empirically the learning rate and momentum values, the congugate gradient technique was used in batch learning mode. Unfortunately, since the training data set is quite big (about 5000

samples), batch learning was slow. For that reason, the conjugate gradient algorithm was applied only to fine tune the training.

The conjugate gradient technique (see Viewgraph 7) is described in [15]. Essentially, it is a technique to determine a set of search directions, which, using a one-dimension line search technique to determine the step size, will very quickly bring you to the function minimum. In batch mode learning, the conjugate gradient technique has been shown to converge much faster than the steepest descent algorithm with line search, which can be viewed as an upper limit for the performance of standard backpropagation [16].

For the steepest descent technique, at a point P_t, the new search direction is the opposite direction pointed to by the gradient, i.e.,

$$d_{t+1} = -\nabla f(P_t) = -\nabla_{t+1} \tag{4}$$

and the new point becomes

$$P_{t+1} = P_t + \lambda_{t+1} d_{t+1} \tag{5}$$

where λ_{t+1} is the step size, which would be user-defined (and constant over many iterations) for the standard BP, or defined by the line search technique used: the line search technique finds λ_t that minimizes the error function f in the search direction [17], pp 25ff.

For the conjugate gradient technique, the search direction is defined by

$$d_{t+1} = -\nabla_{t+1} + \gamma_t d_t \tag{6}$$

where

$$\gamma_t = \frac{\nabla_{t+1} - \nabla_t}{\nabla_t \cdot \nabla_t} \tag{7}$$

is the Polak-Ribiere equation. Note that Eq.5 is still valid and that a line search technique must be used to find λ_{t+1}. In Eq.6, the term γ_t is chosen to insure that the minimum along the previous direction is maintained, i.e., all points along d_{t+1} are minima in regard to the previous direction, obviating the need to search along d_t again (assuming the the function to minimize is quadratic). The new direction is said to be *non-interfering* or *conjugate direction* in regard to the previous one.

2.3 The results

The results are divided into three cases of increasing complexity. In the first case, the number of classes was set to four: Amplitude Modulation (AM), Frequency Modulation (FM), Amplitude Shift Keying (ASK), and Quadrature Phase Shift Keying (QPSK). The classifiers were trained with a data set of 1028 examples. Seventeen features (features 1 to 17 from the complete set) were used to classify the 252 examples of the testing data set, which SNR's were above 8 dB and below 25 dB.

In the second case, the complexity was increased by adding one class, Frequency Shift Keying (FSK), by reducing the SNR to 7 dB rather than 8 dB, and by enlarging the number of features to the complete set of 21. Accordingly, the number of examples in the training and testing data sets were augmented to 3630 and 740 elements respectively.

In the third case, the main objective was to test the algorithm with a SNR as low as 5 dB. The number of features and classes were kept constant, but the number of training and testing examples increased to 4085 and 1000 elements respectively.

Table 2: Percentage of successful classification for the linear classifier and the no-hidden layer BP, using time-domain features.

First	99.2%	99.2%
Second	95.9%	96.9%
Third	93.8%	95.2%

The results presented in Table 2 and in Viewgraph 8 show the overall percentage of successful classification. For the BP, the score represents an average over 2 consecutive consistent runs. Each run was stopped when the cross-validation results started to decrease. About 1600 online iterations were required to train that network. Since the network is very small, it was very sensitive to overtraining, and care was taken to cross-validate regularly. The results show clearly a significant advantgage in using the ANN. Both classifiers have the same number of equations, same number of weights, and therefore the same complexity. The results indicate that the distributions diverge from the normal curve, explaining why the linear classifier was doing consistently worse than the BP as the SNR was diminished. In all cases, the BP was as good as or better than the linear classifier.

The real interest of BP is its ability to map complex decision regions by using hidden layers. The result obtained with the kNN classifier, 99.6%, indicated that the decision regions, in the third case, were not well discriminated by linear equations, and therefore, at least one hidden layer should be used.

Table 3: Percentage of successful classifications for the various BP architectures and the kNN classifier, using time-domain features.

99.6%	99.8%	99.8%	99.75%	99.75%
-	200 batches	150 batches	270 batches	233 batches

As explained in the previous section, using the momentum descent algorithm and keeping the learning rate and momentum constant, the net was trained (online learning) until cross-validation started to decline (about 2000 to 3000 iterations) achieving about 92% of accuracy (up to 98% in some cases). Then,

the conjugate gradient (CG) algorithm was initiated in batch mode, and processed until achieving optimal results conformably to cross-validation (the corresponding numbers of batches are indicated in Table 3). This systematic approach was chosen to avoid having to change empirically the step size and momentum values to achieve optimal performance. On the other hand, using CG all way through, would have been more time consuming.

The question of "optimal architecture" is not completely solved. The results indicated that, although all BPs were able to outperform the kNN, some architectures were slower to converge. In this regards, BP-2 seems to be the best choice. However, since the main concern is to integrate AMR into a real system, the main criteria to consider is the network size, since it dictates the speed at which the classification will be accomplished and the hardware required. In this regard, BP-1 would be the best choice.

3.0 Frequency-Domain Features - PRELIMINARY RESULTS

As discussed earlier, AMR is part of an ESM system, in which the front-end performs energy detection. Since a spectral signal representation is involved for that purpose (such as an FFT or a Periodogram), it would be more optimal from a system design point-of-view to use also this alternate signal representation for AMR. For that reason, AMR using frequency-domain features was investigated under a contract with Dr. S. Haykin and Mr. Nasir Ghani at the Communications Research Laboratory, Hamilton, Ontario.

3.1 The features

There are various spectral representations available that could be used as features for AMR, like: FFT, PSD, Welch's Periodogram, BiSpectrum, Wigner-Ville distribution, Wavelet Transform, Cyclo-Stationary Spectrum, and Chirplet Transform. Each approach has its advantages and disadvantages in respect to the others. In all cases, the output of the preprocessing could be used for signal classification, using an ANN or other classifier. Since the energy detection subsystem (not defined yet) is more likely to use either the PSD or the Periodogram, these have be investigated first.

As in the previous case, a time-window of about 8000 samples (more precisely, 8192 points) was used. The FFT was calculated and the magnitude extracted to obtain the Power Spectrum Density (PSD) of the signal (see Viewgraph 9). Although those 8192 bins could be used to feed the networks, the number of connections involved would be too considerable. Therefore, further preprocessing was required.

Firstly, the number of bins was reduced to 1092 by selecting only those within a bandwidth of 10 kHz around the signal center (the carrier for most modulation type). This 10 kHz bandwidth appeared to be sufficient for the given modulation types. To further reduce the input size, "frequency averaging" was performed: groups of bins were average together. The averages will be referred loosely as outputs of "filters".

Based on an intuitive assumption, since the most critical region should be in the center of the bandwidth, the filters were made narrower in that region, and wider and wider as further away. Rather than performing an exhaustive and very time consuming investigation, the AM/FM case was used to estimated the optimal number of filters (i.e., the number of features) to use. Sets of 10, 20, 30, 60 and 80 were processed and benchmarked using the kNN classifier. Also compared were linear and logarithm scaling when computing the PSD.

It is clear from the results (see Table 4) that increasing the dimensionality of the feature vector at some point leads to performance deterioration. The optimal number is somewhere between 20 and 30 inputs. Note that BP was not trained for each input size since the training is very time consuming, but some tests (10 and 30 inputs) showed very close agreement with the kNN classifier.

Table 4: Performance for the PSD approach, using the kNN classifier for the AM/FM case.

[illegible]	[illegible] Scaling	[illegible] Scaling
10	87.33%	91.17%
20	93.33%	94.67%
30	92.17%	95.50%
60	91.00%	93.50%
80	88.17%	91.00%

A variation to the previous approach, called periodogram, consists of dividing the main window into K sub-windows of M points, where M is a power of two. In our case, since the sub-windows do not overlap, K times M is equal to the main window size, i.e., 8192. Then K spectra are evaluated, one per sub-window, and are averaged together, resulting finally in a M-point PSD.

Finding the optimal combination for K and M is not a trivial problem, and, to be exhaustive, would require multiple simulations. Rather, the AM/FM case and the kNN classifier were once again used to investigate the question. Table 5 summarizes the results.

Table 5: Performance for the Periodogram approach, using the kNN classifier for the AM/FM case.

[illegible]	[illegible]	[illegible] Scaling	[illegible] Scaling
128	64	84.5%	92.2%
64	128	93.0%	94.7%
32	256	97.5%	99.3%
16	512	-	94.7%

The results confirmed the intuitive supposition that logarithmic scaling would be more appropriate, since taking the logarithm enlarges the sidelobe information, which is very helpful in discriminating the modulation types. Also, it appeared that the single-window PSD technique performed less well than the Periodogram technique (M equal to 256 and K equal to 32), which was retained for future investigations.

3.2 The classification process

As previously, the kNN classifier was used to benchmark the performance of a particular data set. We experimentally found that for our application, k equal to one was optimal, which value we have used in the following experiments.

For the ANN, once again a standard BP was used. A slightly modified version of the momentum descent algorithm was implemented on a Warp computer to accelerate the learning process [18]. This Warp computer is a systolic array processor consisting of nine cells capable of 10 Mflops each. The system's new generation, known as iWarp, is now commercially available from Intel: four cells per board of 80 MFlops each, linearly expandable.

The momentum descent algorithm was used to train the networks. The initial weights were set randomly between -0.1 and 0.1, and a different learning rate was chosen for each layer, so that each layer learns at the same rate: the learning rates were smaller than 0.05 and inversely proportional to the layer fan-in [19]. Also, to accelarate the learning and optimize the results, we periodically decreased the momentum, starting with a big value (around 0.9). Training was considered completed when crossvalidation results started to decrease.

Once again, one and two-hidden layer networks were investigated. In the first case, the number of hidden nodes was set to 80, 90, 110, 140, and 160 in order to find the optimal architecture. Note that there are 70 inputs to the nets and 9 outputs (See Viewgraph 10). For the two-hidden layer case, only two architectures were tested: (70, 50, 30, 9) and (70, 80, 30, 9). Further explorations will be required to find the best architecture, if the results show any advantage using two hidden layers.

3.3 The results

Using the periodogram preprocessing, a data bank was created for nine modulation types: AM, FM, ASK, QPSK, non-coherent FSK, coherent FSK, Binary Phase Shift Keying, Single-Side-Band Upper-Band, and Single-Side-Band Lower-Band. The data set was divided into three groups: training, testing, and crossvalidation, consisting of, respectively, 1800, 300, and 100 examples per class, 70 features per example. Note that the SNR was set to vary between 5 dB and 25 dB.

For the single-hidden layer networks, the best performance was achieved when using twice as many hidden neurons as inputs (refer to Table 6 and Viewgraph 11). Note also, that the BP outperformed by 1% the kNN (97.8%). The training process for the networks required between 2,300 and 2,800 iterations. For the two-hidden layer case (see Table 7), the results were very slightly inferior. However, the number of connections was significantly smaller and the convergence significantly faster: between 1,400 and 1,800 iterations. This indicates that the decision regions are possibly non-convex, maybe disjoint, and would be more easily separated with two-hidden layers.

Table 6: Performance of the single-hidden layer BP using the Periodogram preprocessing.

Hidden Units	Total Weights	Successful Classification
80	6409	98.26%
90	7209	97.73%
110	8809	98.22%
140	11,209	98.74%
160	12,809	97.96%

Table 7: Performance of the two-hidden layer BP using the Periodogram preprocessing.

Hidden Layer 1	Hidden Layer 2	Total Weights	Successful Classification %
50	30	5359	98.33%
80	30	8389	98.26%

4.0 Conclusion

The problem of AMR has been investigated for many years using time-invariant features. However, the results obtained were not fully satisfactory for an ESM purpose. It was proposed to increase the robustness of the system in two ways: by improving the set of features, and by using a classifier able to map complex decision regions. The results showed that both ways contributed in improving the system accuracy. On the one hand, using the classic probabilistic linear classifier on the new set of features provided impressive results in the order of 94% for signal SNR's as low as 5 dB. On the other hand, the performance was further improved up to 99.8% using an ANN.

The results are very encouraging and further work will be conducted toward the same direction (see Viewgraph 12). First, the number of modulation types will be increased to 10. Then, feature selection will investigated: how would the system performance be affected by removing some features; which features are the most/least significant. Those questions will be investigated using the kNN classifier and possibly the genetic algorithm [20]. Also, another type of classifier will be investigated: it is proposed to test a binary decision tree classifier on the data set. This type of classifier has proven its ability to solve complex problems very successfully.

A new approach has also been investigated. We have shown that with frequency-domain features only, it is possible for an ANN to recognize the various modulation types. Two frequency representations were investigated: PSD and Periodogram. It is planned to investigate a third signal representation known as BiSpectrum. Also, the Optimal Brain Damage algorithm [21] will be implemented on the BP simulator. A preliminary test indicated that the algorithm could removed about 40% of connections while maintaining the accuracy.

But clearly the major future milestone will be to test those approaches with real signals. Although care was taken in designing the simulator that generated the data bank, real signals will create probably more complex decision regions. Although the performance is expected to decrease, hopefully the ANN and the feature extractor will be robust enough to provide acceptable performance, which would have not been achievable with the classifiers used typically for AMR.

5.0 References

[1] S. FAULKNER, "CERES - A Computer-Aided Radio Monitoring and Surveillance System", International Defence Review: An Intervia Publication, June 1977

[2] Derek ELSAESSER and Brian KOZMINCHUK, "Advanced Communication ESM System (ACES) Design Proposal", DREO TN, 1991

[3] Locus Inc, "MRD-100B Operational Manual", Locus Inc., P.O. Box 740, PA 16804, May 1988

[4] Watkins-Johnson, "WJ-9040 MRU109 Modulation Recognizer", Technical Data, C E T Division, Watkins-Johnson, December 1990

[5] René LAMONTAGNE, "Envelope and Instantaneous Frequency Estimation of Narrowband Modulated Signals (U)", DREO TN, 1991

[6] René Lamontagne, "Modulation Recognition: An Overview", DREO TN 91-3, March 1991

[7] Richard O. DUDA and Peter E. HART, "Pattern Classification and Scene Analysis", John Wiley, 1973, Chap. 2

[8] K. FUKUNAGA, "Introduction to Statistical Pattern Recognition", Academic Press, 1972, pp 178

[9] I.D. LONGSTAFF and J.F. CROSS, "A Pattern Recognition Approach To Understanding The Multi-Layer Perceptron", Royal Signals and Radar Establishment, Memorandum 3936, July 1986

[10] Richard P. LIPPMANN, "Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987

[11] Yoh-Han PAO, "Adaptive Pattern Recognition and Neural Networks", Addison-Wesley, 1989

[12] T.P. KUDRYCKI, " Neural Network Implementation of a Medical Diagnosis Expert System", Master's Thesis, College of Engineering, University of Cincinnati, 1988

[13] Alianna J. MAREN, Craig T. HARSTON, and Robert M. PAP, "Handbook of Neural Computing Applications", Academic Press, 1990

[14] Y. LEE, S.-H. OH, and M.W. KIM, "The Effect of Initial Weights on Premature Saturation in Back-Propagation Learning", Internat. Joint Conf. on Neural Networks, Vol.I, 1991

[15] W.H. PRESS, B.P. FLANNERY, S.A. TEUKOLSKY, and W.T. VETTERLING, "Numerical Recipes in C", Cambridge University Press, 1988

[16] Etienne BARNARD, "Optimization for Training Neural Nets", IEEE Trans. on Neural Networks, Vol.3, No.2, March 1992

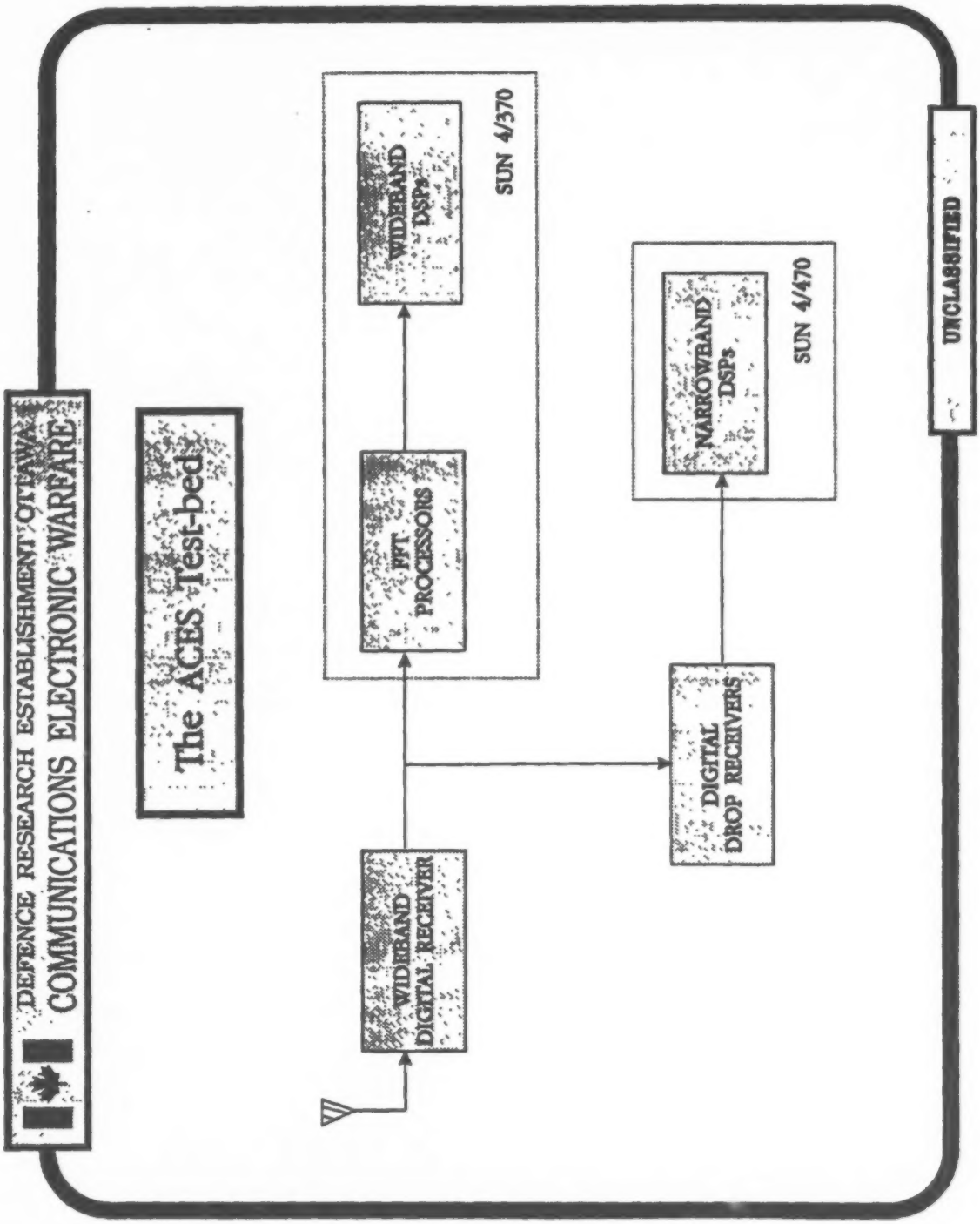
[17] R. FLETCHER, "Practical Methods for Optimization", John Wiley, 1980

[18] D.A. POMERLEAU, G.L. GUSCIORA, D.S. TOURETZKY, and H.T. KUNG, "Neural Networks Simulation at Warp Speed: How We Got 17 Million Connections per Second", IEEE Internation Conference on Neural Networks, Vol. II, 1988

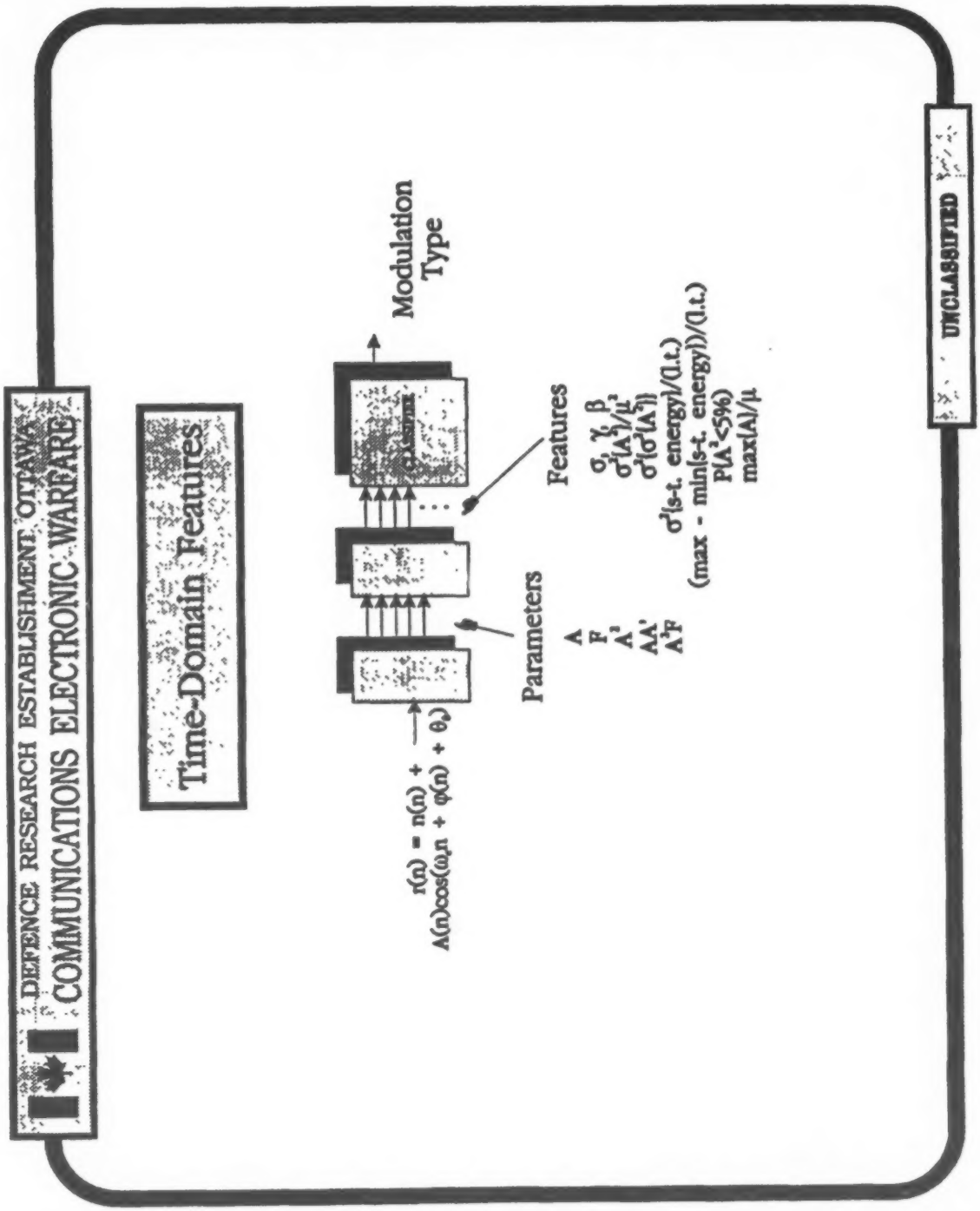
[19] Anthony P. RUSSO, "Feedforward Neural Network Design", Conference on Neural Networks for Ocean Engineering, Washington, D.C., August 1991

[20] Frank Z. BRILL, Donald E. BROWN, and Worthy N. MARTIN, "Fast Genetic Selection of Features for Neural Networks Classifiers", IEEE Trans. on Neural Networks, Vol. 3, No. 2, March 1992

[21] Yann LE CUN, John S. DENKER, and Sara A. SOLLA, "Optimal Brain Damage", Advances in Neural Information Processing System 2, ed. David S. TOURETZKY, 1990



Viewgraph 3: The ACES Test-bed



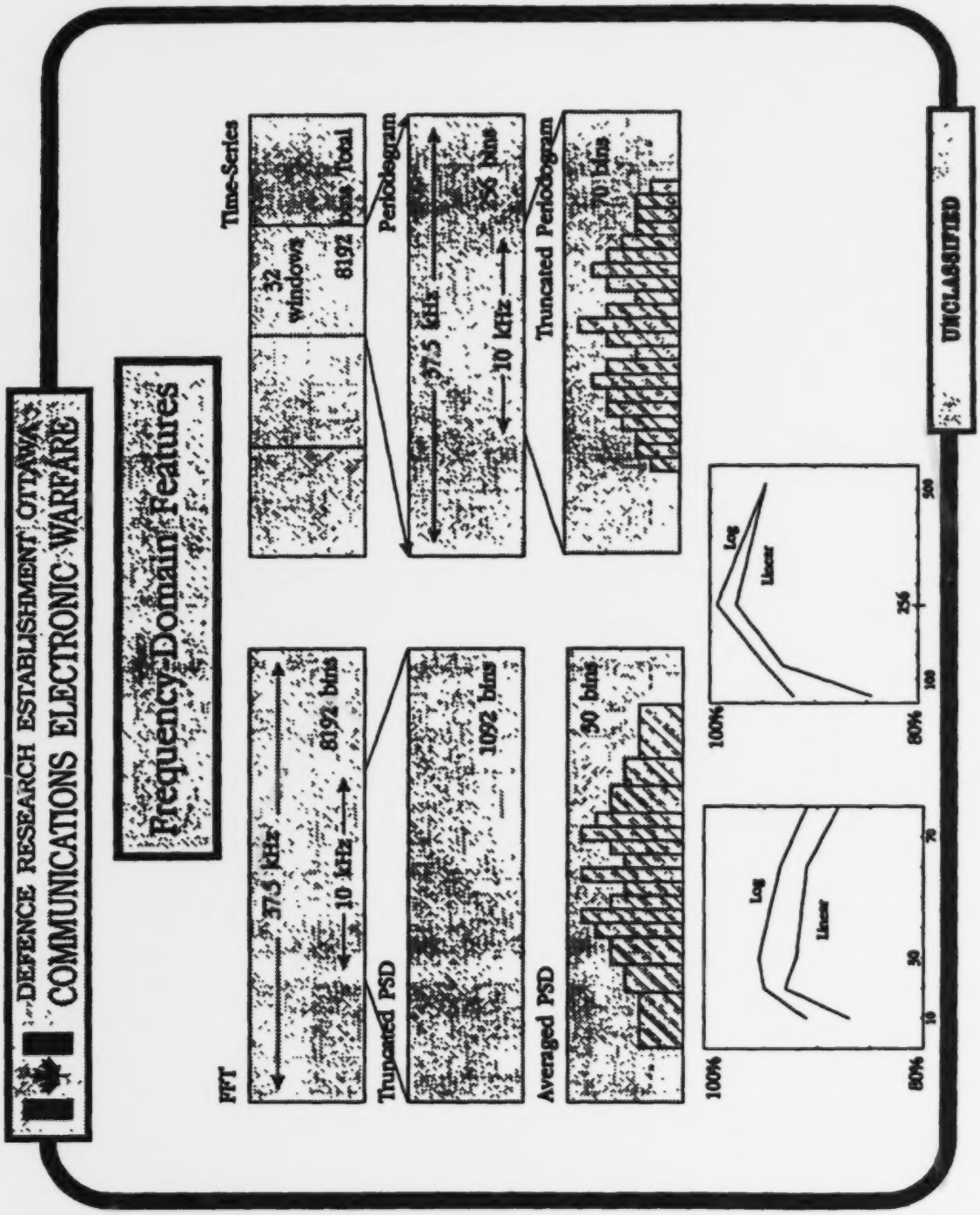
Viewgraph 4: Time-Domain Features

Viewgraph 5: Time-Domain: Classification

Viewgraph 6: 2-Hidden Layers Architecture

Viewgraph 7: Conjugate Gradient Direction

Viewgraph 8: Time-Domain: Preliminary Results



Viewgraph 9: Frequency-Domain Features

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
COMMUNICATIONS ELECTRONIC WARFARE
Frequency-Domain: Classification
Classifier:
kNN
Euclidean distance
optimal for k = 1
BP
momentum descent algorithm
Warp computer: 9 cells of 10 MFlops
random initial weights (0,1)
fixed learning rate, momentum < 0.9
one-hidden layer: 80, 90, 110, 140 and 160 hidden units
two-hidden layers: (70, 50, 30, 9) and (70, 80, 30, 9)
Data Bank:
9 classes: AM, FM, ASK, QPSK, NC-FSK, CPFSK, BPSK, USB, LSB
70 features: Periodogram
5 dB < SNR < 25 dB
training data set: 1800 examples per class
testing data set: 300 examples per class
validation data set: 100 examples per class
η = ce/fan-in
fan-in
UNCLASSIFIED

Viewgraph 10: Frequency-Domain: Classification

Viewgraph 11: Frequency-Domain: Preliminary Results

Viewgraph 12: Conclusion

#13:303 #5

Classification of Passive Sonar Signals Using a Backpropagation Neural Network: Simulation Studies

Tim R.H. Cutmore
and
G. Robert Arrabito

Defence and Civil Institute of Environmental Medicine

Abstract

For the past several decades passive sonar signals have been processed by the human operator using visual and/or auditory displays to achieve detection and classification of target vessels. Recently, a small number of studies have begun to apply neural network techniques to this domain of signal processing. Time series data presents challenges to these pattern recognition networks and preprocessing methods are often critical. In this paper, a passive sonar hydrophone simulation is outlined. Methods and results of applying the backpropagation algorithm with a 3-layer feedforward network to simulated passive sonar data are presented. In particular, different types of fourier preprocessing and network parameters are examined for their effects on convergence rate (learning speed) and classification performance. Finally, the extension of this work to real passive sonar data and potential pitfalls are discussed.

1. Introduction

1.1 Background

The detection and classification of underwater acoustic sources are difficult and important problems, (Urick, 1983). The target sources may be masked by oceanic background noise and multiple targets may be present simultaneously. Classification, which assumes detection, has additional challenges of correct generalization and discrimination. For the past several decades passive sonar signals have been processed by the human-operator using visual and/or auditory displays to achieve detection and classification of target vessels.

Although the use of neural networks for processing sonar data is relatively recent, a large number of studies have been published for active sonar. These have concerned themselves with navigation of autonomous underwater vehicles (e.g., Shazeer et al., 1991; DeMuth et al., 1990; Schiller et al., 1989), classification of undersea objects (Gorman et al., 1988; Castellano et al., 1990; Porto, 1989; Venugopal et al., 1990) including mine classification (Bello, 1991). General views from a military perspective can be found in Webster (1991) and Hewish (1990). From a more esoteric perspective, a few studies have begun to examine mammalian echolocation with neural networks (e.g., Roitblat et al., 1989; in dolphins and Chen et al. 1988 in bats).

Work on passive sonar processing has been much less intensive. A recent search by the

authors of several scientific data bases turned up only a few studies. Jacobson et al., (1988) conducted a preliminary study with a feedforward network trained by backpropagation to classify a set of natural acoustic sources derived from actual hydrophones. Various network configurations were examined and most were able to discriminate the target sources from each other and background noise. Liou (1988) using simulated data obtained partial success in classifying ship parameters, such as propeller shaft speed and number of propeller blades, with multiple sources present. Baran et al., (1991) used similar methods with natural data from hydrophone recordings of acoustic signals emitted by ships and was able to train a network to discriminate ship types. Bridle et al., (1990) applied a neural network in a sonar line tracking task.

1.2 Objectives

This report describes the initial steps towards complementary objectives: 1) to build a model of a natural acoustic source and 2) to develop in parallel a neural network architecture capable of discriminating features of this source. In building a model of an acoustic type complexity may be added by degrees to more closely approximate the natural source. This would permit a systematic study of the dimensions of the problem. For example, in the natural domain a single acoustic source may yield a variety of waveforms due to doppler effects and variation in the hydro-acoustic medium. Noise sources may mask the signal. In addition, several sources may be active simultaneously and this would require solving an acoustic occlusion problem.

In our selection of an acoustic source type it was of interest to compare machine verses human ability to classify sounds. To this end we have used a set of sources which were used by McFadden et al., (1992) in a study of how human subjects are able to classify acoustic sources (hereafter called "targets") and their associated "speeds". The quotations are used to indicate that these sources were simulated. The sound sources are described below.

1.3 Sound Simulation and Preprocessing

Figure 1 shows a system view. Sounds were synthesized using the CARL Cmusic software package (CARL, 1985). The McFadden et al., (1992) study used 10 target sources which

were defined by a set of frequency components blended together. Three instances of each target were constructed by varying a "pulsing" component (simulated cavitation) which was interpreted to indicate the speed of the target. The sounds were presented in the auditory modality only (i.e., no visual displays were used).

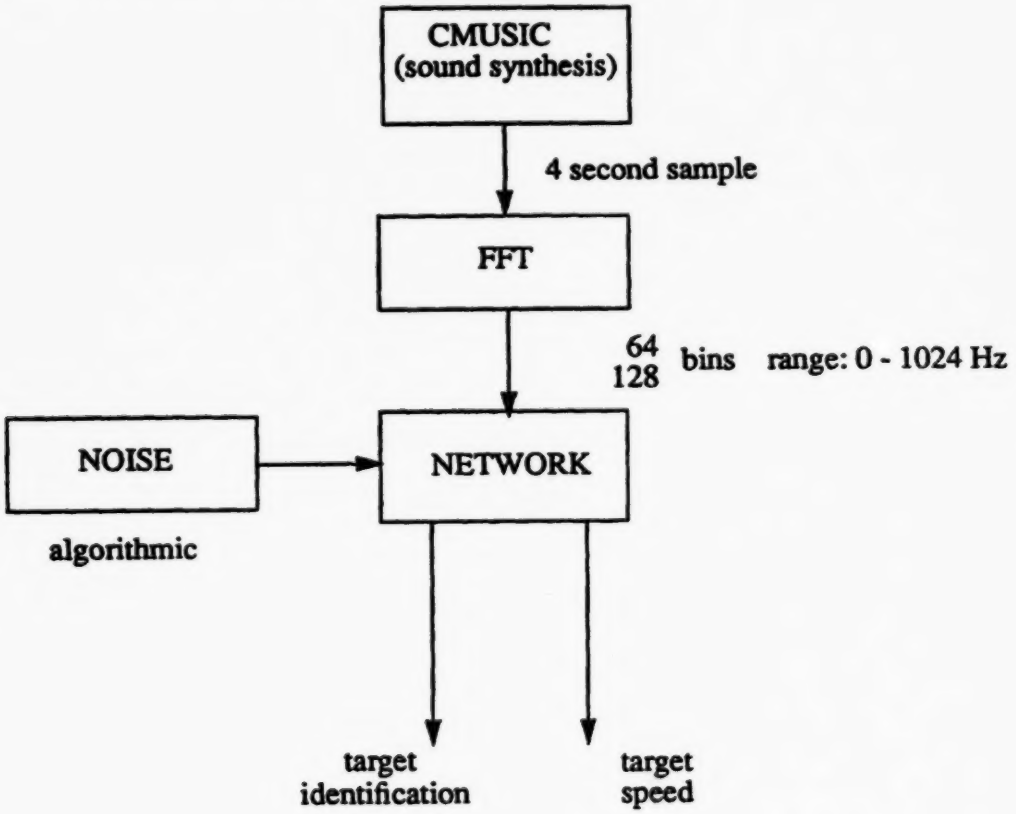


Figure 1

The sounds were passed through a Fourier Transform (FFT) to yield a power spectrum (as amplitude squared values) using a 4 sec sample at a sampling rate of 8192 Hz. Only frequencies below 1024 Hz were then retained, since none of the sounds produced power above this frequency. Two resolutions were examined, 8 Hz (128 power bins) and 16 Hz (64 power bins). Figure 2 shows examples of the 128 bin power spectra for each of the 10 targets.

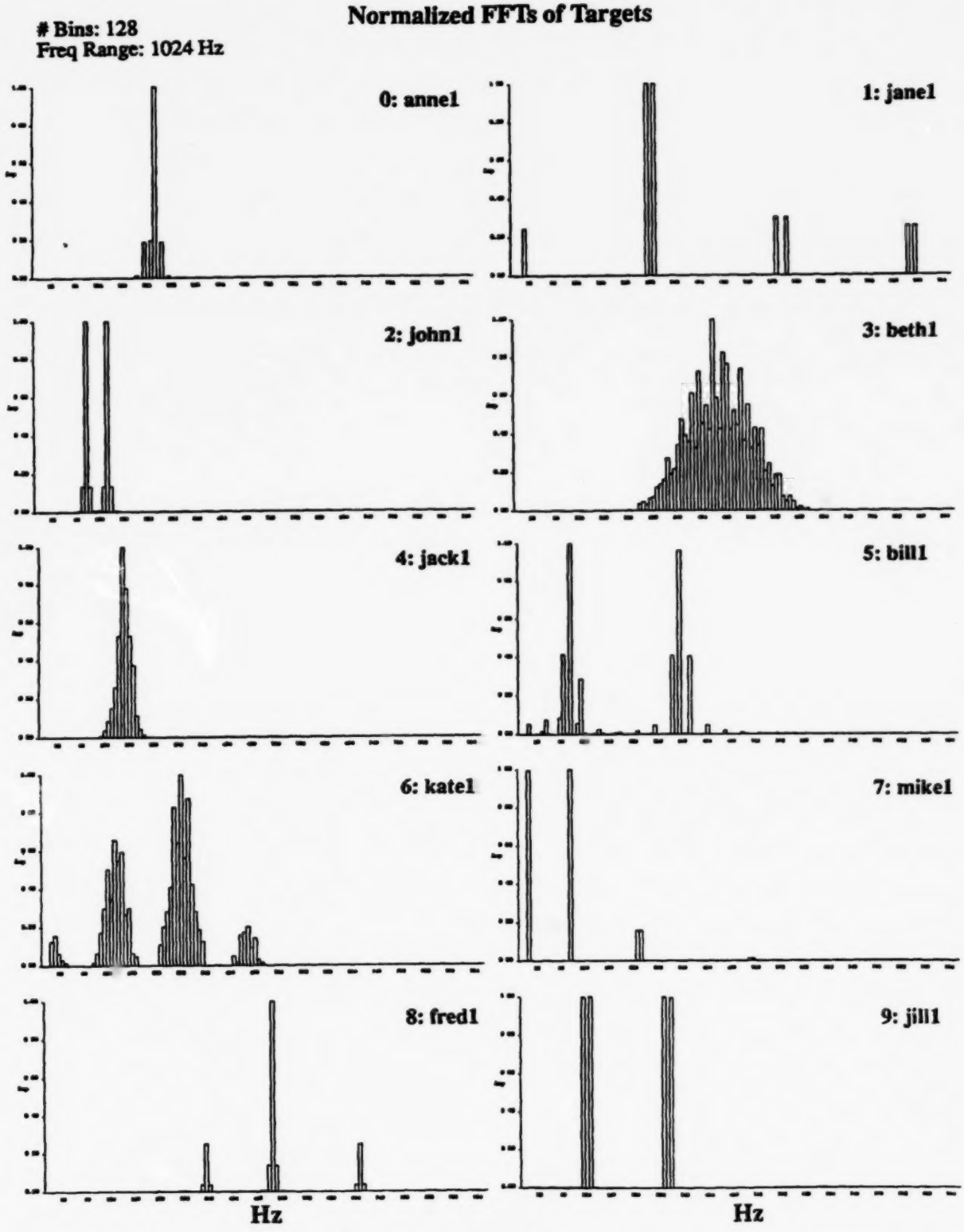

Normalized FFTs of Targets
Bins: 128
Freq Range: 1024 Hz
0: anne1
1: jane1
2: john1
3: beth1
4: jack1
5: bill1
6: kate1
7: mike1
8: fred1
9: jill1
Hz
Hz

Figure 2

Figure 3 illustrates how the instances could vary for a given target, depending on the speed. Both simple additive low frequency components (pulses) and amplitude modulated cavitation was included. Twenty sounds (two from each target) were used to train the subjects (and the networks here) with the remaining 10 used to test generalization.

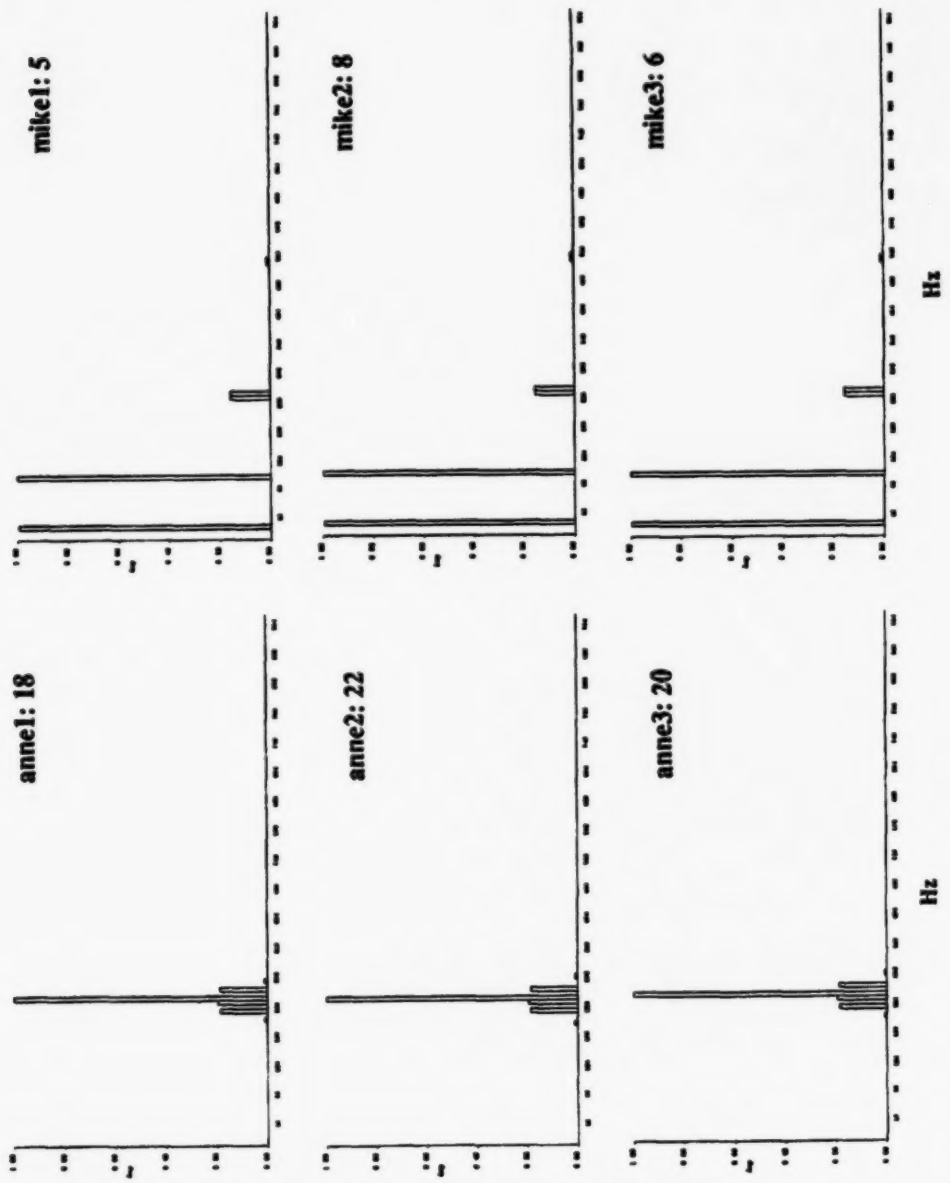


Figure 3

1.4 Network Selection

In selecting a simple feedforward network trained with backpropagation to perform the classification and speed estimation, its well studied structure, learning rule and associated parameters (momentum, temperature, learning rate) were decisive factors. If it should turn out that this extensively used network was inadequate for the task, then at least a baseline level of performance would be established for any future work with more elaborate networks. Other types of networks have been used to classify sonar signals. Genetic algorithms have been used to classify sonar images (Montana et al., 1989). Specht (1989) used an exponential in place of the sigmoid function to produce a "probabilistic network" which apparently can be trained much more rapidly than by back propagation. Good results were obtained with a sonar submarine detection task.

Figure 4 shows the network architecture that was used for both experiments. The inputs were the FFT normalized power bins. Various numbers of hidden nodes were used. Fifteen outputs were divided into two sets. One set was used to train the network as a classifier, with 10 alternatives (one for each target). The second set was used to train the network to produce a binary representation of the speed of the targets. A threshold was applied to both sets to determine whether an output node was "on" or "off". The activation function was a sigmoid with a range of -1 to +1. The generalized delta rule was used to train the network with a momentum term (Rumelhart et al., 1986).

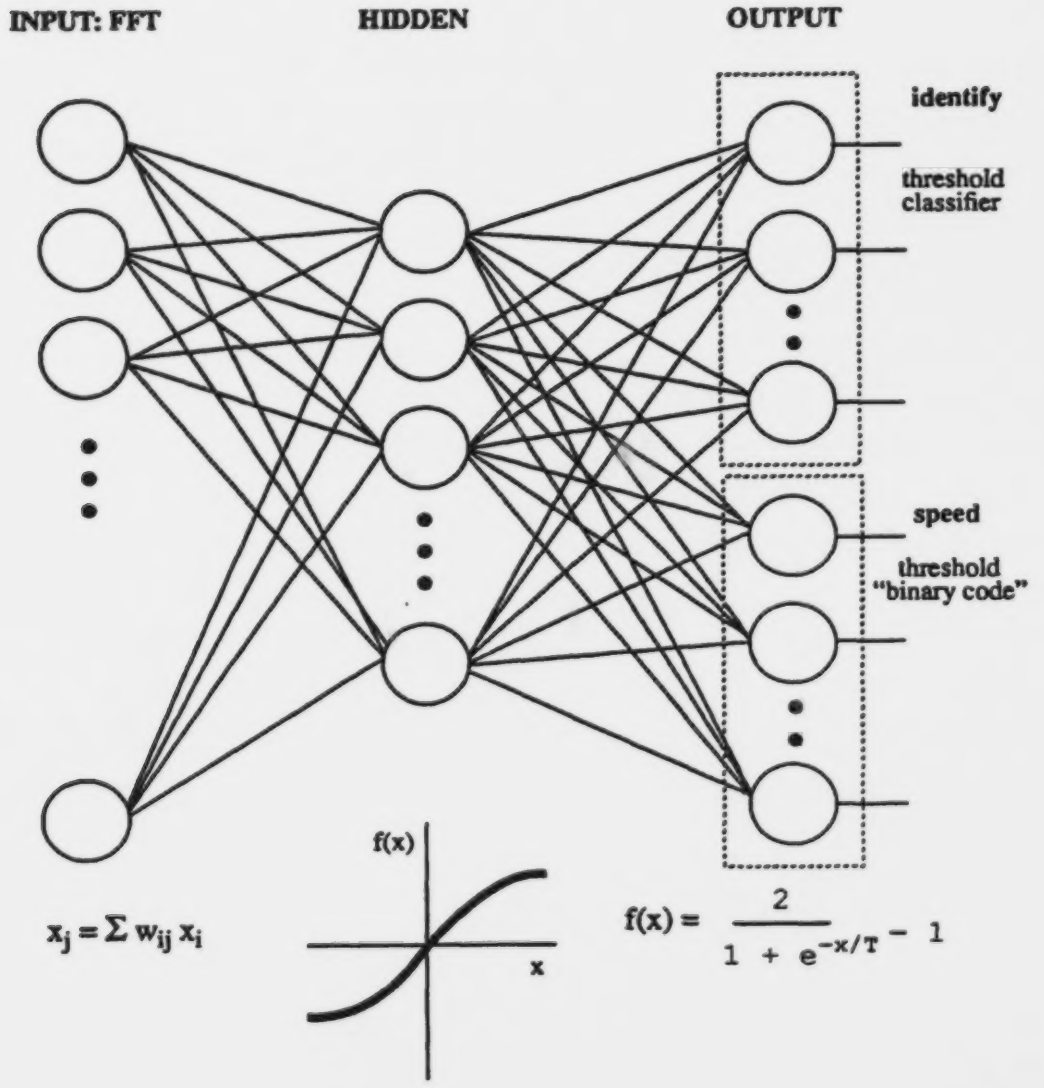
2. Experiment 1

The purpose of the first experiment was to examine a variety of parameters with "clean" (no noise) target signals in an attempt to find the "best" networks for solving the problem.

2.1 Network Designs

Figure 5 shows the parameters which were fixed and systematically varied. The fixed parameter levels were selected during a series of pilot trials. For example, it was found that an initial learning rate of 0.3 which was permitted to increase in proportion (0.4) to the change in RMS error at the output nodes, produced rapid, yet stable convergence. Bias inputs were generally not found to assist in convergence and so were left "off". Finally, a fixed number of cycles was selected at 2048, since it was found that in virtually all cases, convergence (if it was going to occur at all) was essentially complete at this time. The network was always trained by epoch. In other words, the full set of training patterns was presented before weights were updated.

FEEDFORWARD NEURAL NETWORK



$\Delta w = \lambda \delta x + \mu \delta w$

$\delta = (t - o) f'$ (output layer)

$\delta_i = (\Sigma w_{ij} \delta_j) f'$ (hidden layer)

Figure 4

FIXED PARAMETERS:

1) Initial learning rate: $\lambda = 0.3$

2) Adaptive learning: $a = 0.4$

3) Bias Nodes: 0.0

4) Cycles: 2048

VARIABLE PARMATERS:

1) Temperature: T = 0.5, 1.0, 4.0

2) Momentum: $\mu = 0.0, 0.3, 0.9$

3) FFT Input Resolution: 8, 16 Hz (0 - 1024 Hz)

4) Hidden Nodes: 0, 4, 8, 16, 32

Figure 5

The range of the variable parameters was also selected following pilot work to assess their general efficacy. For example, extending the temperature beyond 4.0 produced similar results albeit with more learning cycles required. Temperatures below 0.5 produced very unstable behaviour. For momentum, values above 0.9 may produce unstable behaviour as well. Five variations of network were examined which differed according to the number of hidden units: none (perceptron), 4, 8, 16 and 32. These will be referred to as xH where x is the number of hidden nodes. As previously noted 8 Hz and 16 Hz FFT resolutions were used. The experiment thus included: three levels of temperature, three levels of momentum, two input resolutions and five hidden node alternatives. This produced 90 different network configurations. For each configuration, five replications were averaged, each having a different set of random initial weights. These weights were constrained between -0.5 and +0.5.

2.2 Results

The two measures of primary interest were percent of targets correctly identified and the error of speed estimation. The RMS error and learning rate measures receive brief comment.

The primary result of interest for this first experiment was the rapidity with which a given network could achieve high performance for identifying a target and estimating its speed, both for trained targets and for ones to which it had not yet been exposed. Therefore, the learning curves are shown to illustrate network behaviour.

Since there are a very large number of network configurations, illustrative sets are provided with some parameters ignored. These are divided into topics for emphasis: training and testing, temperature and momentum, and 16 verses 32 hidden nodes. Finally, 64 FFT bin inputs showed similar yet frequently slightly poorer performance for all measures, particularly for the speed estimation. For simplicity only the results with the 128 bin networks are shown.

2.2.1 RMS Error and Learning Rate: Figure 6 shows examples of the relationship between RMS error and learning rate. These (as well as all remaining graphs) show log linear plots of training cycles and a performance measure. The perceptron and 16 hidden node networks are shown with each of the T and μ levels. The RMS error for the test patterns are also shown. In general, networks were able to converge to an RMS error below 12% for at least some parameter combinations and the learning rate increased in proportion to this. However, as shown for the 16H network, T = 0.5 did not produce good performance. Finally, test patterns generally showed a minimum RMS error well above training patterns, at about 0.5 to 0.6.

2.2.2 Training and Testing: Figure 7 shows the results of training and testing for target identification for the 0H, 8H and 16H networks. Only $\mu = 0$ results are shown. The following findings were of interest here: 1) Each network has one or more parameter combinations that result in 100% correct target identification and the number of cycles required is exceptionally small - 4 to 64 cycles. This is the case for both training and test patterns as they nearly overlay each other. 2) The perceptron converges much more rapidly than networks with hidden nodes; 3) While the perceptron's training and test performance appears to be simply shifted to larger numbers of learning cycles as T increases, networks with hidden nodes have an optimal temperature.

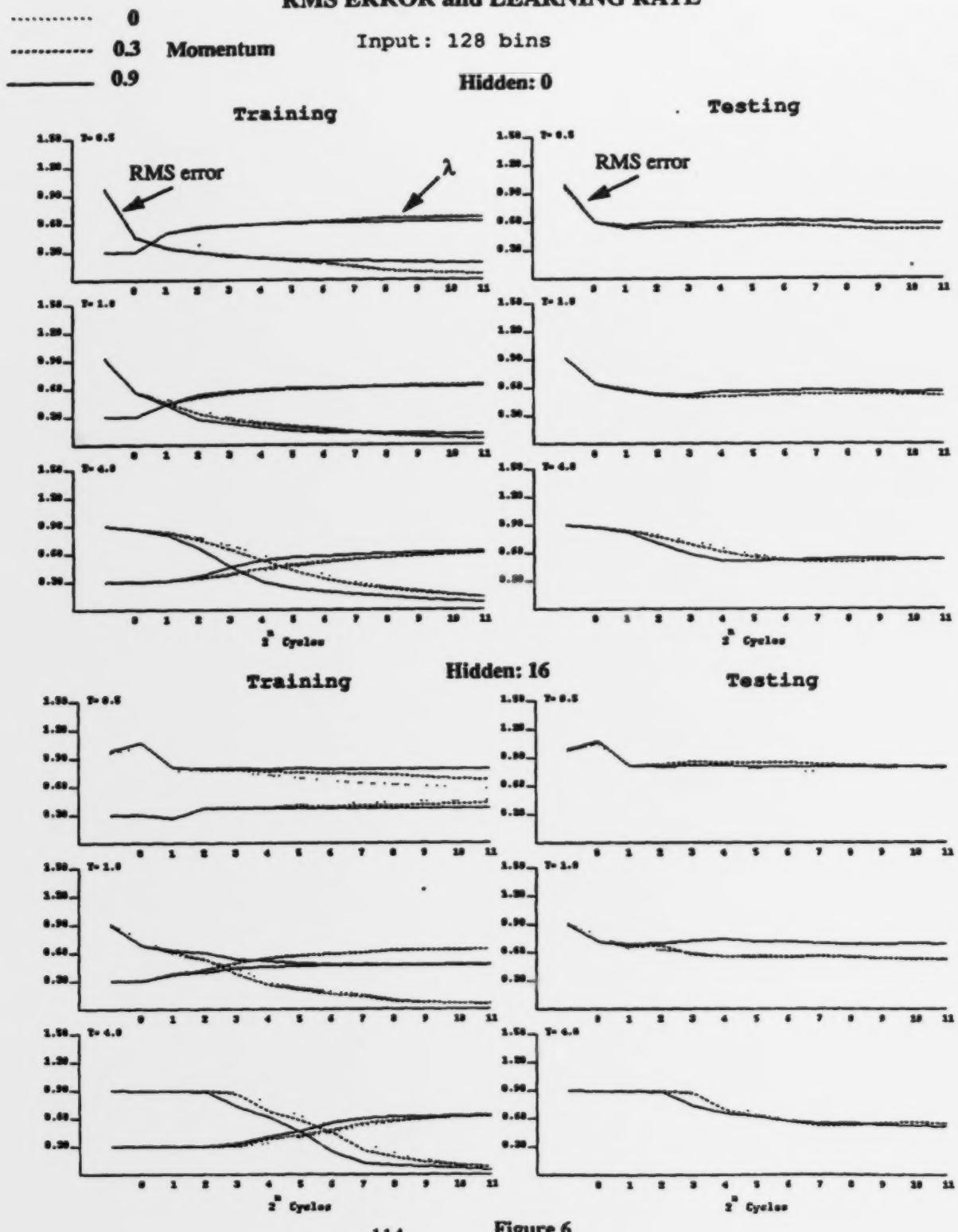

RMS ERROR and LEARNING RATE
0
0.3 Momentum
0.9
Input: 128 bins
Hidden: 0
Training
Testing
RMS error
λ
RMS error
Hidden: 16
Training
Testing
2^n Cycles
2^n Cycles

Figure 6

TRAINING and TESTING

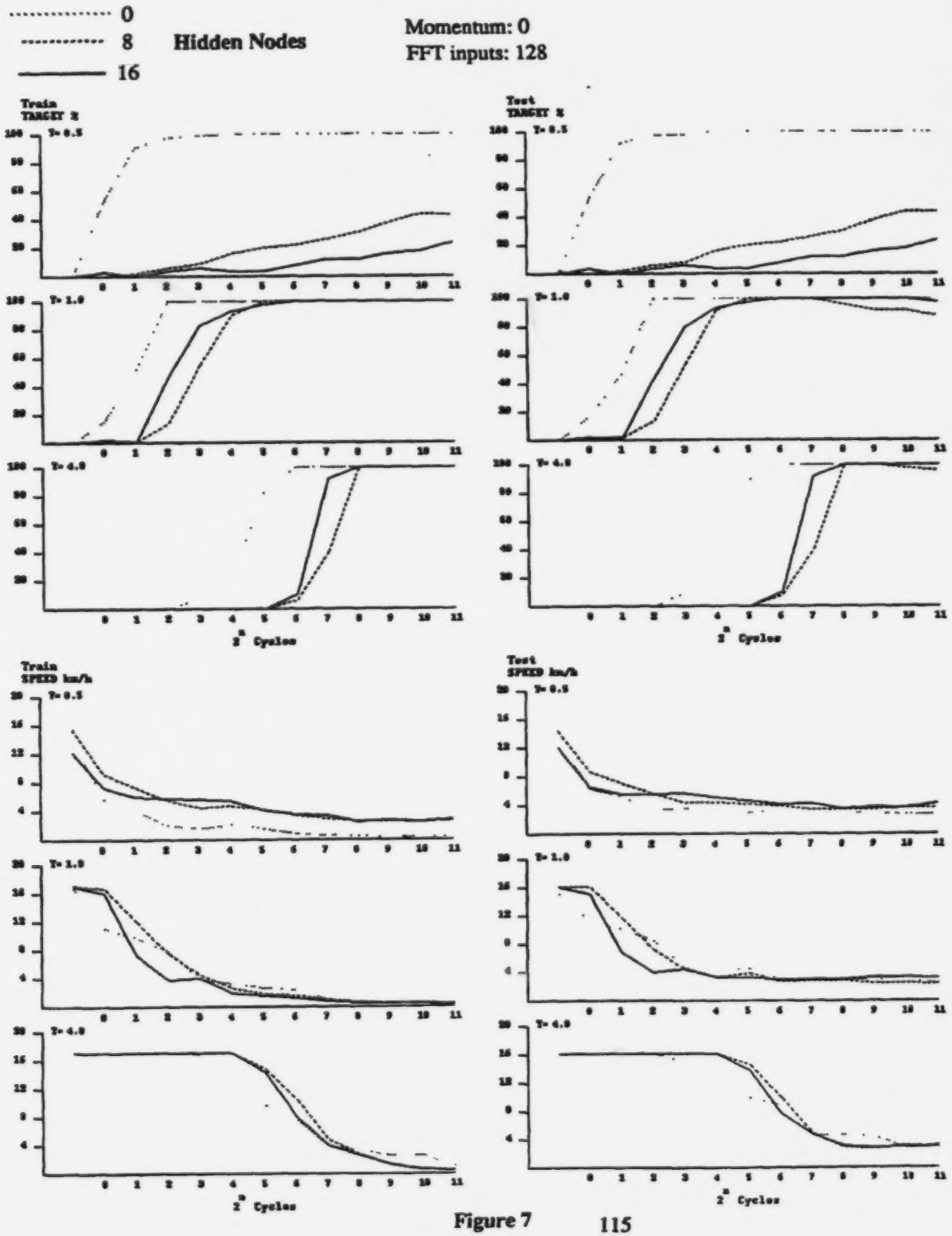


Figure 7

Figure 7 also shows the results for the speed estimation error. The results of interest are: 1) The error in the trained patterns approaches very near to zero, while the test patterns appear to yield at best a speed error of 4 units. 2) The perceptron is not so clearly the fastest learning network with this measure. Indeed at T = 1.0 the training and test performance of the 16H net was found to be equal to or better than all other networks. 3) A comparison of Figs 7 and 8 shows the speed estimation error requires more learning trials than target identification to achieve highly accurate performance for both training and test patterns. 4) An overtraining effect is evident for the 8H network at T = 1.0.

2.2.3 Temperature and Momentum: Figure 8 shows the effects of varying both temperature and momentum. From these data the "best" T and μ parameters were selected for each network class[1]. For the perceptron these were: T = 1.0; μ = 0.9; For the 4H network, T = 4.0, m = 0.9; For the 8H, 16H and 32H (not shown) networks, T = 1.0, μ = 0.3.

Figures 9 shows results which address the question of the utility of including 32 hidden nodes over 16 hidden nodes. All T and μ levels are shown. It appears that these networks perform very similarly for both training and test patterns. A slight advantage for the 32 node network appears for T = 4.0. Therefore, there may be little advantage to using more than 16 hidden nodes to solve the identification and speed estimation problems here.

2.3 Conclusions:

Trained Patterns

1. A network with two sets of outputs was able to learn both target classifications and speed estimates with very high accuracy. For some networks all 10 target id's were learned in as little as 4 iterations. In all networks, speed estimates took more iterations for good performance.

2. On both these measures, the network equals or outperforms human subjects who were trained to make auditory discriminations (McFadden, 1992). For target id Humans have shown a mean accuracy score of 65%, although one person did score 100%. Most of the network configurations learned target identification to 100% accuracy of classification.

1. The target test and both speed estimate results showed a similar pattern of results.

TEMPERATURE and MOMENTUM

FFT inputs: 128 (Training)

..... 0

..... 0.3 Momentum

——— 0.9

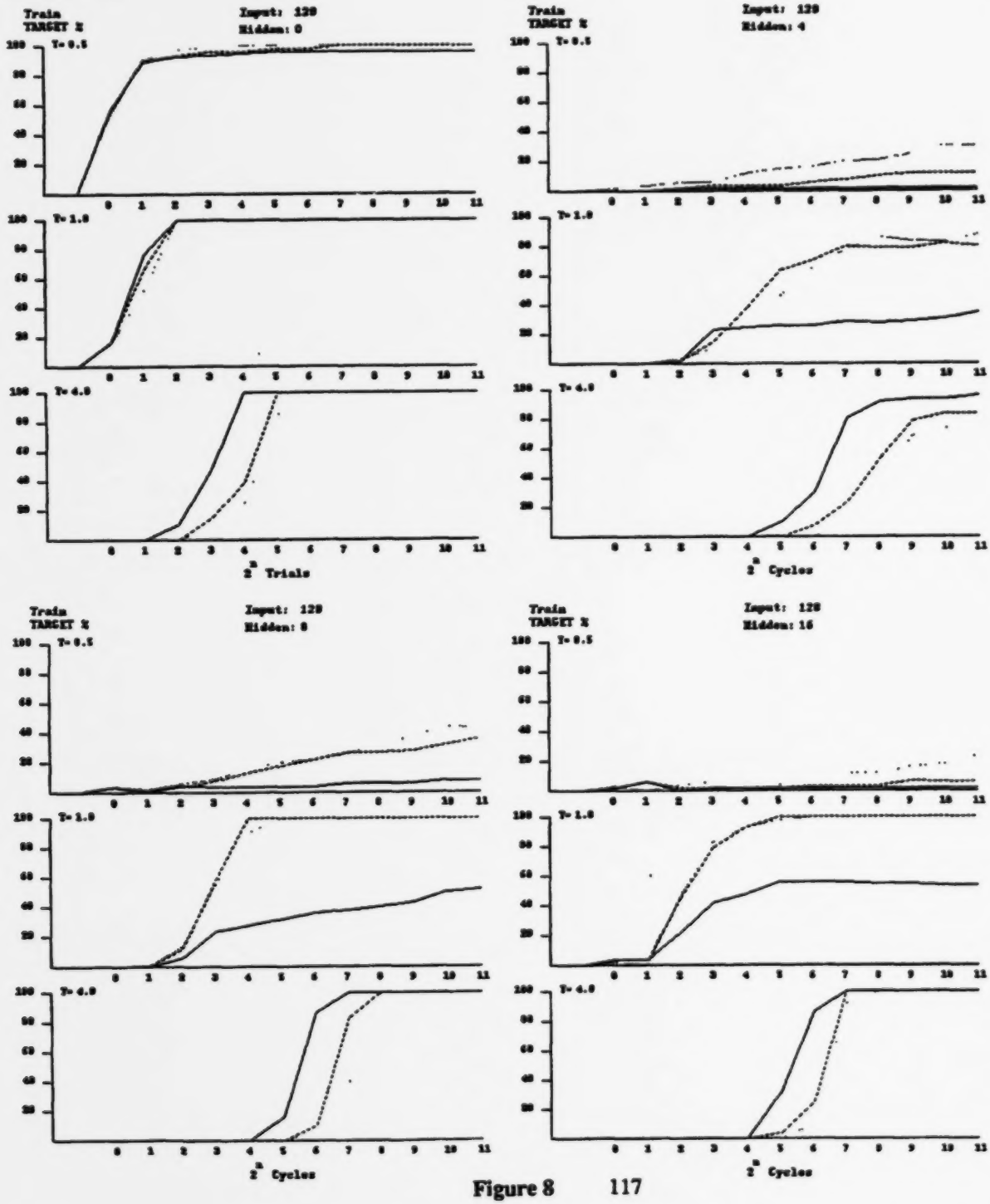
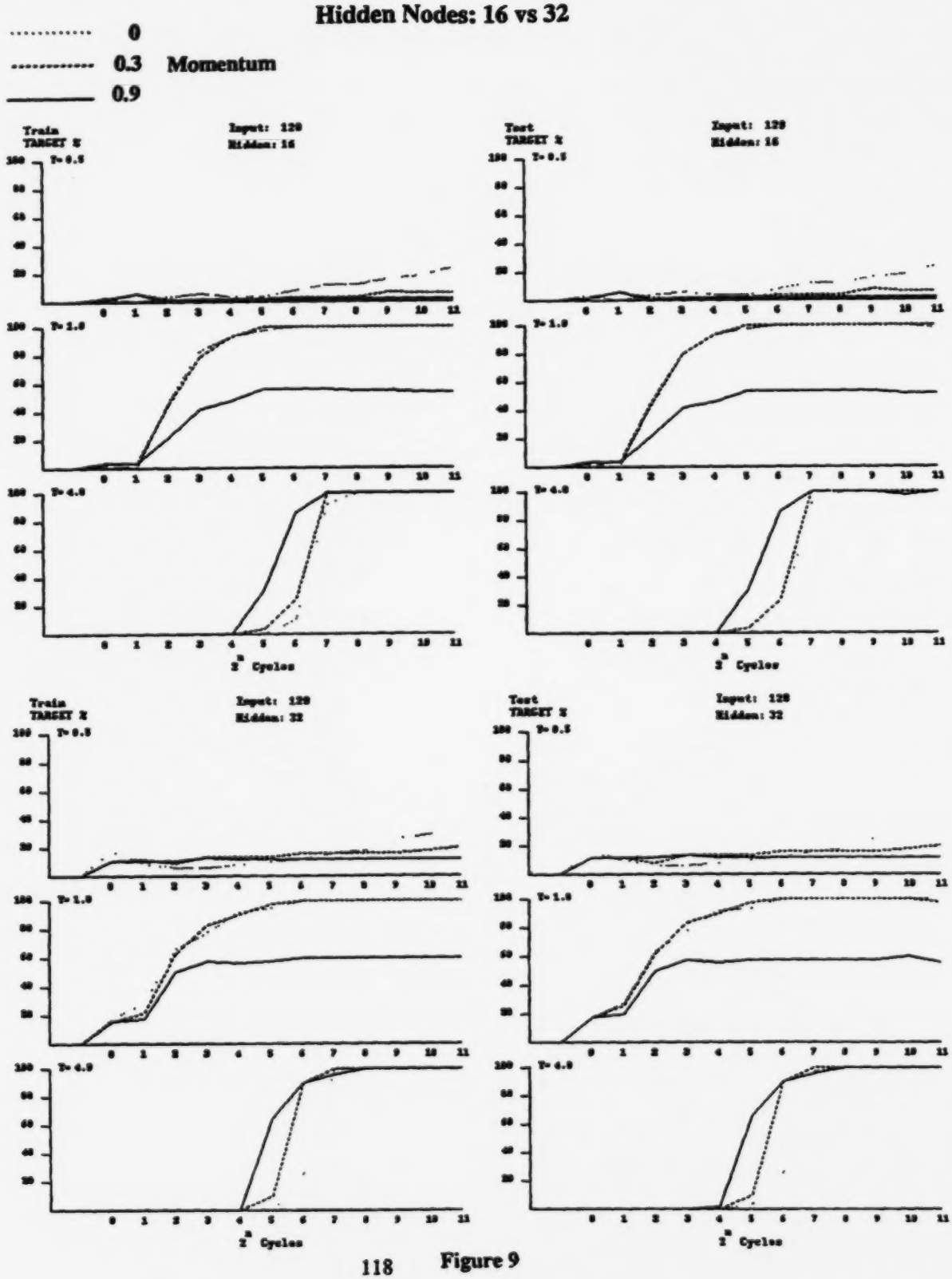


Figure 8 117


Hidden Nodes: 16 vs 32
0
0.3 Momentum
0.9
Train
TARGET %
Input: 128
Hidden: 16
Test
TARGET %
Input: 128
Hidden: 16
Input: 128
Hidden: 32
Cycles

118 Figure 9

For the speed estimation error, humans learned to estimate to a mean score of 4 units. Most network configurations were able to learn this estimate to less than 1 unit.

Test Patterns (same target id's, different speeds)

3. Several networks classified targets (is) with perfect accuracy. Speed estimates were accurate to within 4 units for many networks.

4. On both measures, networks performed similarly to human subjects. For target id, mean performance for human subjects was at 65%, with the best scoring at 96%. Networks, in general, disseminated the test targets as well as the trained targets achieving 100% accuracy. For speed estimation, humans and networks scored similarly with an error of about 4 units.

Hidden Nodes

5. Observing the high performance of the perceptron the conclusion can be asserted that for a simple input space use a simple network (i.e., perhaps a perceptron will do). But as the input space become more complex, one or two hidden layers may become necessary.

6. Observing the difficulty with which the 4H network learned, it may be that such a network may have suffered from "overcompression" of information. However, it would be of interest to increase the learning cycles to determine whether better performance could be achieved.

Input Resolution

7. Fourier bin resolution of 8 Hz performed a little better than 16 Hz for resolving speed differences.

3. Experiment 2

Upon achieving the encouraging results in experiment 1 with clean signals it was of interest to examine the performance of the networks given a noisy input signal. This would improve the simulation model of natural target sounds in the ocean which has a broadband noise spectrum (Urick, 1983).

3.1 Noise Generation

Gaussian noise was digitized from a noise generator and added to the activation of the input nodes. The broad band (0 - 1024 Hz) signal to noise ratio (SNR) used was -5 db on aver-

age (since the normalized target signals varied in summed power the SNR's ranged from 2 db to -8 db). Two examples are shown in figure 10. A new noise pattern was generated each time a pattern was presented to the network. Two conditions were examined: 1) The noise was simply added to the inputs and the 10 classifier and 5 speed estimation nodes were used to train the network. 2) Everything in case 1 was repeated but in addition, an 11th classifier node was added to the output set and a novel noise pattern was presented to the network for each epoch of training (i.e. each cycle) and a unique noise pattern alone was presented for each test. Thus, the noise in both conditions was truly unique in every circumstance. The speed of a noise pattern was trained to zero.

These noise conditions addressed the issue of whether a feedforward network could handle both random variation in the input patterns as well as the question of whether the network could learn to recognize noise (or discriminate it) when it occurred in isolation.

3.2 Networks Used

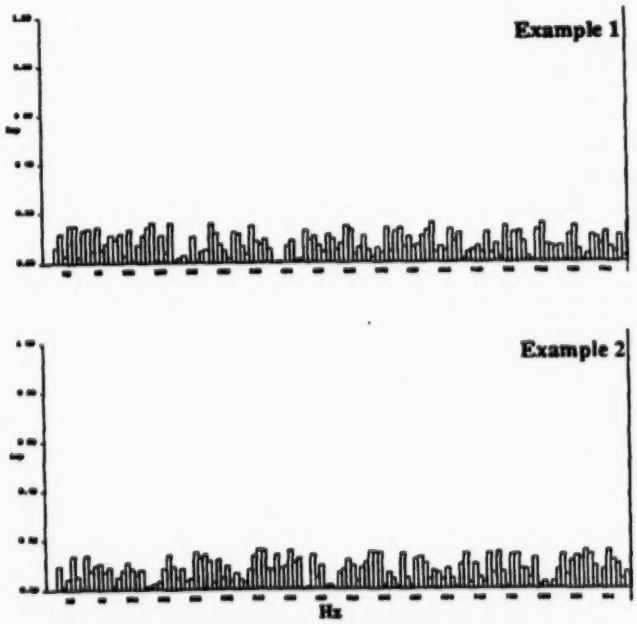
From experiment 1, five networks (0H to 32H) were selected on the basis of best all around performance on classification and speed estimation. For 0H, $T = 0.9$; $\mu = 1.0$; 4H, $T = 4.0$; $\mu = 0.3$; for 8H, 16H and 32H, $T = 1.0$ and $\mu = 0.3$. In all cases the 128 node FFT input was used. The training and test pattern sets of experiment 1 were used for comparison to the clean target signal results. Again, five replications were averaged starting with different initial weights in each case. Finally, pilot tests with the noisy signals revealed oscillatory RMS error for the learning and adaptive parameters used in experiment 1. It was found that performance tended to be more stable for a low λ value. Therefore it was fixed at 0.1 with no adaptation.

3.3 Results

3.3.1 Target identification: Figure 11 shows the results for the five networks. The solid line is from experiment 1 for reference. The perceptron appears to have a notable problem for both noise conditions. For the noise added condition it does not achieve 100% target identification for either the training of test patterns. It does even more poorly for the noise discrimination condition. Since the graphs appear to reach asymptote after 512 trials, the perceptron may be incapable of learning to handle the noisy input patterns. The 4H network also had considerable difficulty. However, the figure suggests that, at least for training patterns, that continued

training beyond 2048 cycles may result in better performance, since the scores do not reach asymptote. The 8H, 16H and 32H networks performed progressively better. All three found solutions to achieve 100% recognition of trained patterns. Although the 8H and 16H networks achieved 100% test pattern recognition, this did not stabilize. The 8H network appeared to be particularly sensitive to over training as the decline in the noise added condition shows. Only the 32 node network learned both trained and test patterns to a stable 100% accuracy. In fact, this network even outperformed its processing of clean signals when noise was added to the patterns. This network achieved 100% accuracy after only 16 cycles with the noisy signals, but required 64 cycles with the clean signals.

SPECTRUM OF NOISE ADDED TO TARGET SIGNALS



Broad Band Signal to Noise Ratio: -5 db
Peak Single Bin Signal to Noise Ratio: 10 * log(1.0 / 0.2) = 7 db

Figure 10

This page intentionally left blank.

NOISE ADDED TO TARGET SIGNALS
TARGET IDENTIFICATION

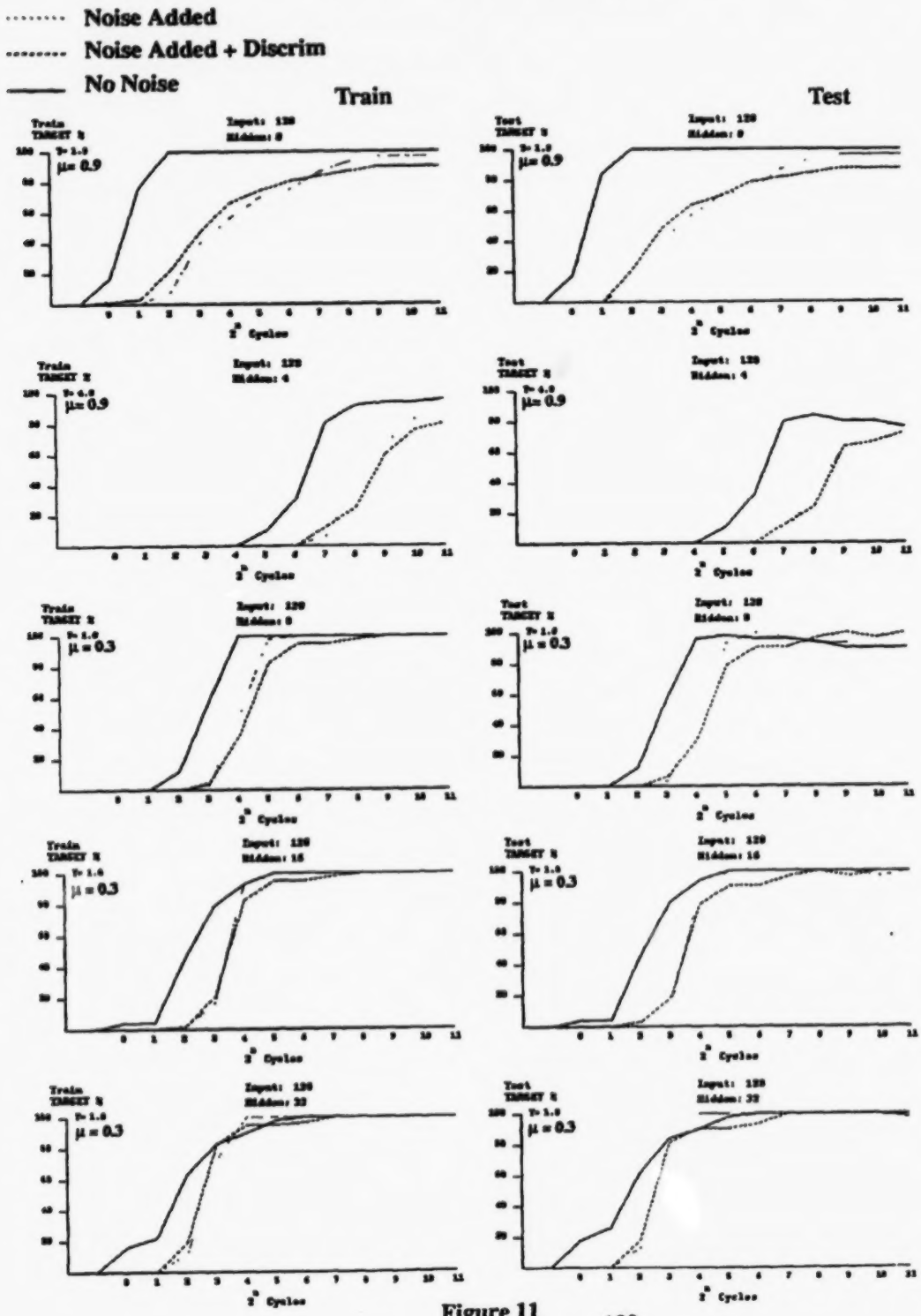


Figure 11

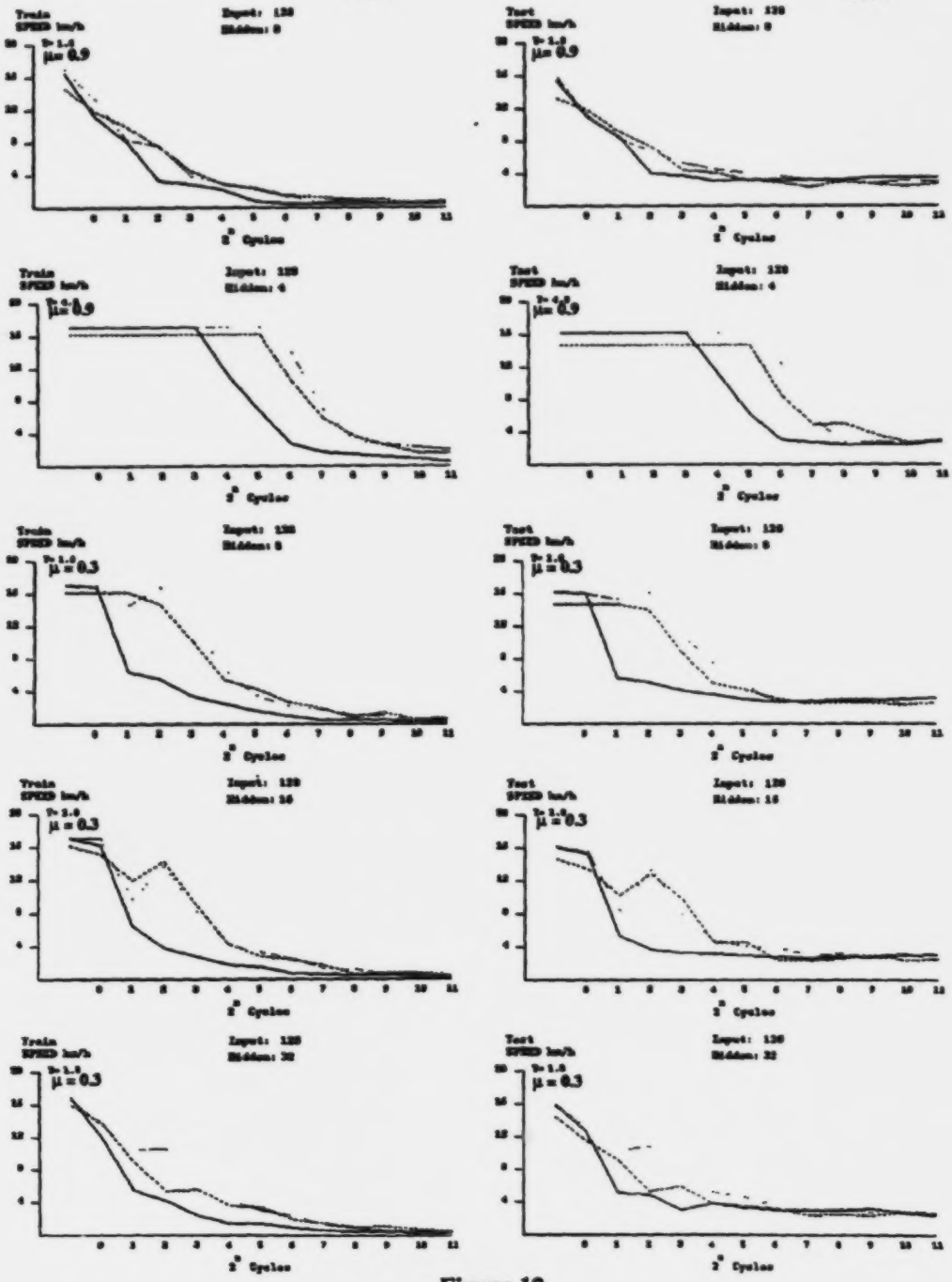
NOISE ADDED TO TARGET SIGNALS
SPEED ESTIMATION

..... Noise Added

..... Noise Added + Discrim

——— No Noise

Train Test



124 Figure 12

References

Baran, R.H. & Coughlin, J.P. (1991) A neural network for target classification using passive sonar. Proceedings of the International Society for Optical Engineering, 1471 (pp164-76).

Bello, M. (1991) Concepts for the use of three-dimensional depth maps in underwater mine classification. Report No. CSDL-R-2303, Charles Starck Draper Lab. Inc: Cambridge (51p)

Bridle, J. & Smith, D (1990) Line tracking using an artificial neural network and a dual process model Baysian mathematical framework. In: Colloquium on the Role of Image Processing in Defence and Military Electronics.

Computer Audio Research Laboratories (1985) CARL Startup Kit. University of California, San Diego.

Chen, L. & Brennan, C. (1988) Preliminary to a neural network model of sonar-based target discrimination in the echolocating bat. Tech Report, Providence, RI. Dept. Psychology(46p).

DeMuth, G. & Springsteen, S. (1990) Obstacle avoidance using neural networks. In: Proceedings of the Symposium on Autonomous Underwater Vehicle Technology, (pp213-215).

Gorman, R.P & Sejnowski, T.J. (1988) Analysis of hidden units in a layered network trained to classify sonar targets. Neural Networks, 1, (pp 75-89).

Hewish, M. (1990) Neural computing: Bringing brain power to defence. International Defence Review, 23: 2, (pp 17-19).

Jacobson, J.Z., Laplante, M. & Mogford, R.H. (1988) Artificial neural networks in naval warfare. Research Report 2-6803, Operations Research and Analysis Establishment.

Liou, C. (1988) Design of neural networks to classify sonar targets. In: International Neural Network Society (p 34).

McFadden, S.M., Belanger, J.M. & Yundt, K.A. (in preparation) Defence and Civil Institute of Environmental Medicine.

Montana, D.J. & Davis, L. (1989) Training feedforward neural networks using genetic algorithms. Proceedings of the 11th International Joint Conference on Artificial Intelligence, (762-7).

Porto, V.W. (1989) Pattern recognition of undersea objects using a neural network. In: 23rd Annual Asilomar Conference on Signals, Systems and Computers, (pp 376-380).

Roitblat, H.L., Moore, P.W.B., Nachtigall, P.E., Penner, R.H. & Au, W.W.L. (1989) Dolphin echolocation: Identification of returning echoes using a counterpropagation network. In: International Joint Conference on Neural Networks, (pp 295-300).

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986) Learning internal representations by error propagation. In: Parallel Distributed Processing Eds. D.E Rumelhart & J. L. McClelland. Cambridge: MIT Press.

Schiller, I. & Tench, K.A. (1989) A neural-network-based autonomous underwater vehicle guidance system. Proceedings of the 6th International Symposium on Unmanned Untethered Submersible Technology, (pp 312-19).

Shazeer, D.J. & Bello, M. (1991) Detection and classification of undersea objects using multilayer perceptrons. In: Applications of Artificial Neural Networks II (pp 622-636).

Specht, D. F. (1989) Probabilistic neural networks (a one-pass learning method) and potential applications. In: Wescon /89 Conference Record, Electronic Conventions Inc. (pp 780-5).

Urick, R.J. (1983) Principles of Underwater Sound, New York: McGraw-Hill.

Venugopal, K.P., Pandya, A.S. & Sudhakar, R. (1990) Sonar target recognition using neural networks, 3rd International Workshop. Neural Networks and Their Applications, (pp 563-565).

Webster, W.P. (1991) Artificial neural networks and their application to weapons. Naval Engineers Journal, 103, (pp 46-59).

This page intentionally left blank.

Multi-sensor data fusion for target aquisition
CRAD SIPWG
Workshop on Neural Networks

Rony Philippe

May 1992

In the past few years, DREV has undertaken the study and development of intelligent systems for application in area defence weapons (ADW), intelligent mines, battlefield surveillance systems and peacekeeping missions. The current problem is to be able to acquire and identify noncooperative vehicles using a number of sensors (e.g. geophones, microphones and electromagnetic sensors). By making use of a host of signals, the aim is to provide as fine a level of discrimination as possible. This approach, commonly referred to as *data fusion*, is an area of interest in many applications and very diverse disciplines.

In a first step, it is necessary to determine the signal features which will serve to discriminate between the vehicles. Once an appropriate set of features has been selected, the neural net is a proposed method for classifying and/or identifying the various signatures, and come to a decision on the identity of the unknown vehicle. The neural network should prove well suited for the expected variability of the signatures of vehicles due to the noise and uncertainties of a combat environment.

Currently at DREV, in parallel with the search for appropriate signal features, work is about to begin in the simulation of neural nets by first repeating some previously published results to gain a practical understanding of the artificial neural net, study the various network topologies and the various training algorithms.

Among the simulations to be run are:

1. character recognition: given an array of pixels, a neural net is trained on a standard alphabet of characters, and then tested for recognition of corrupted versions of these characters;

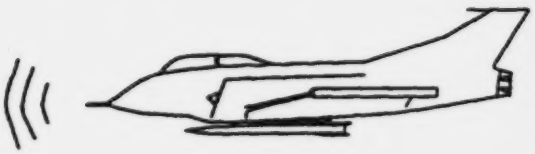
2. adaptive antennas: a phased-array antenna controlled by a neural net is made to adaptively produce a peak and a null in its radiation pattern in specific directions.

Expertise gained in performing these simulations and others will be useful in designing a neural net for the classification/identification of vehicle signatures.

Data Fusion for Vehicle Identification

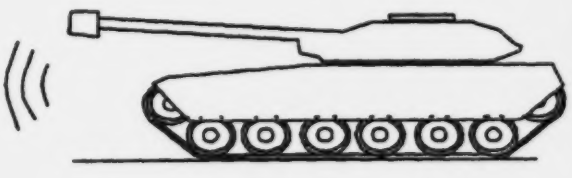
Aim:

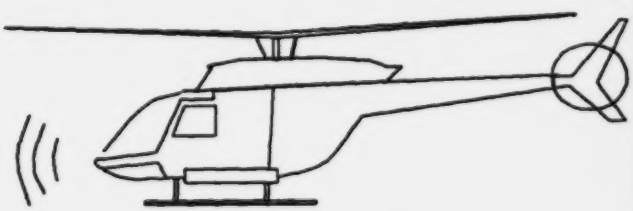
Acquire/identify vehicles using a multi-sensor system.



Signatures of interest:

- Acoustic emissions
- Seismic emissions
- Electric emissions
- Magnetic emissions



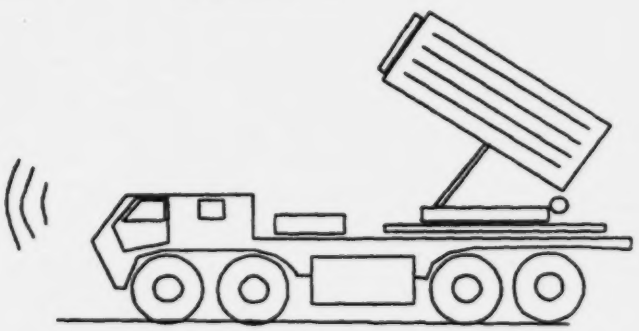


Characteristics:

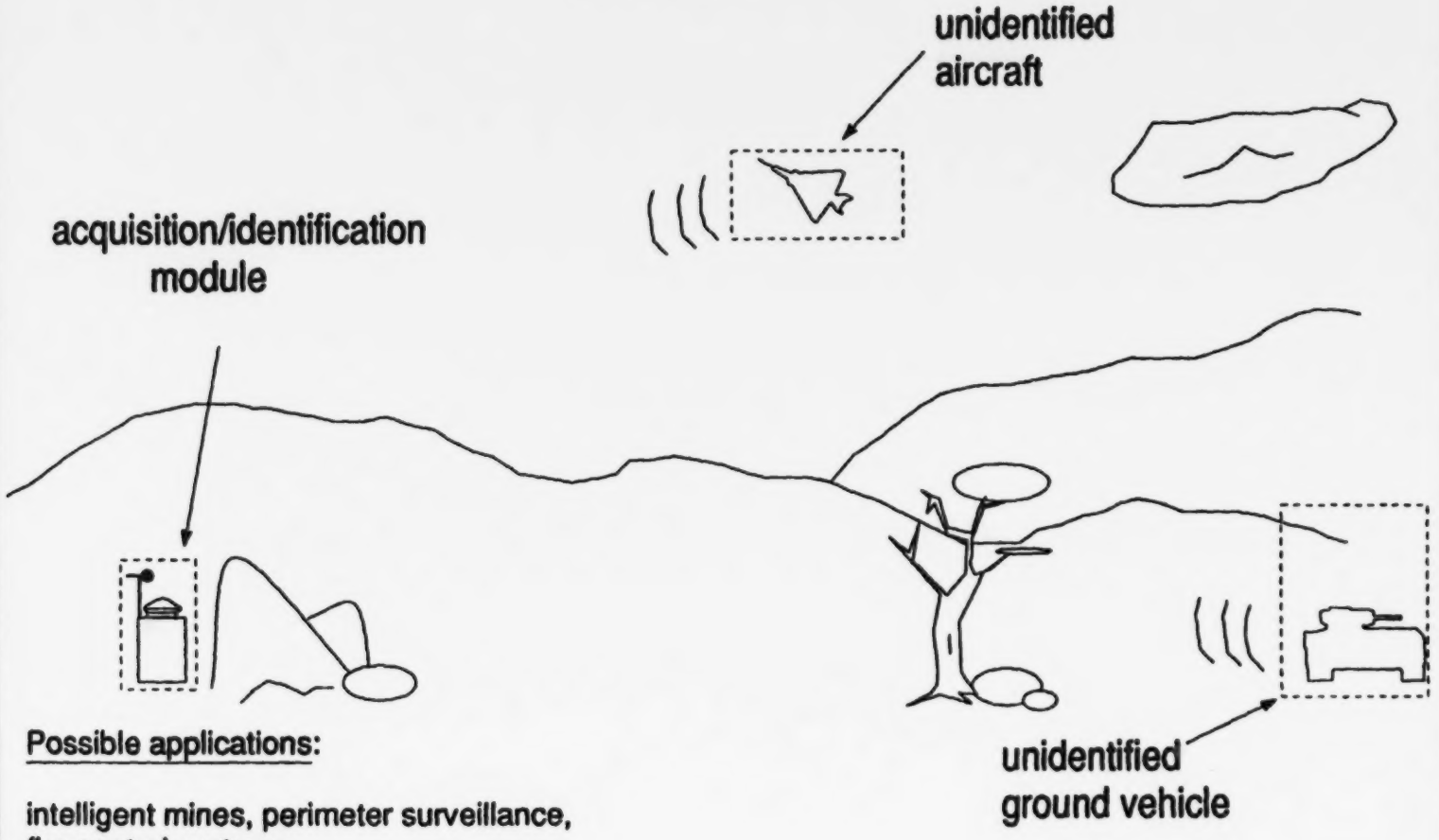
- passive detection
- non-imaging processing

Vehicle groups:

- winged aircraft
- tracked vehicle
- helicopter
- wheeled vehicle



Example scenario



Possible applications:

intelligent mines, perimeter surveillance,
fire control systems.

Initial project focus: single vehicle with running engine.

Vehicle identification process

1) Selection of discriminating features:

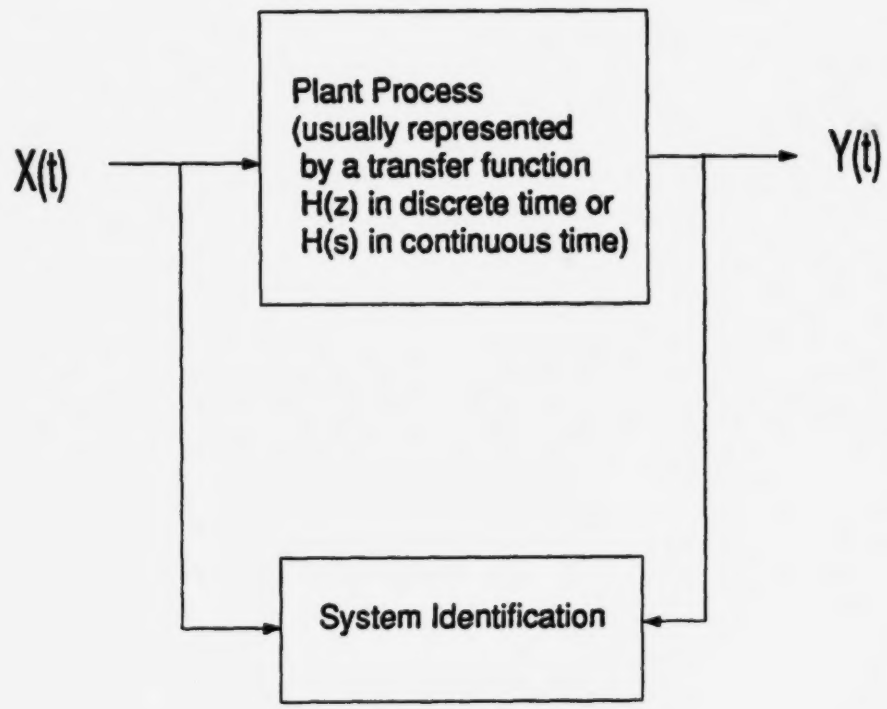
signature gathering, signal/system modeling.

2) Classification of those features:

developing a systematic process by which the observed signal features are to be classified in order to arrive at a decision about the identity of the unknown vehicle.

Among the suggested solutions are expert systems, fuzzy logic and artificial neural networks. The latter will be the first to be investigated as a solution to the problem of classifying vehicle signatures.

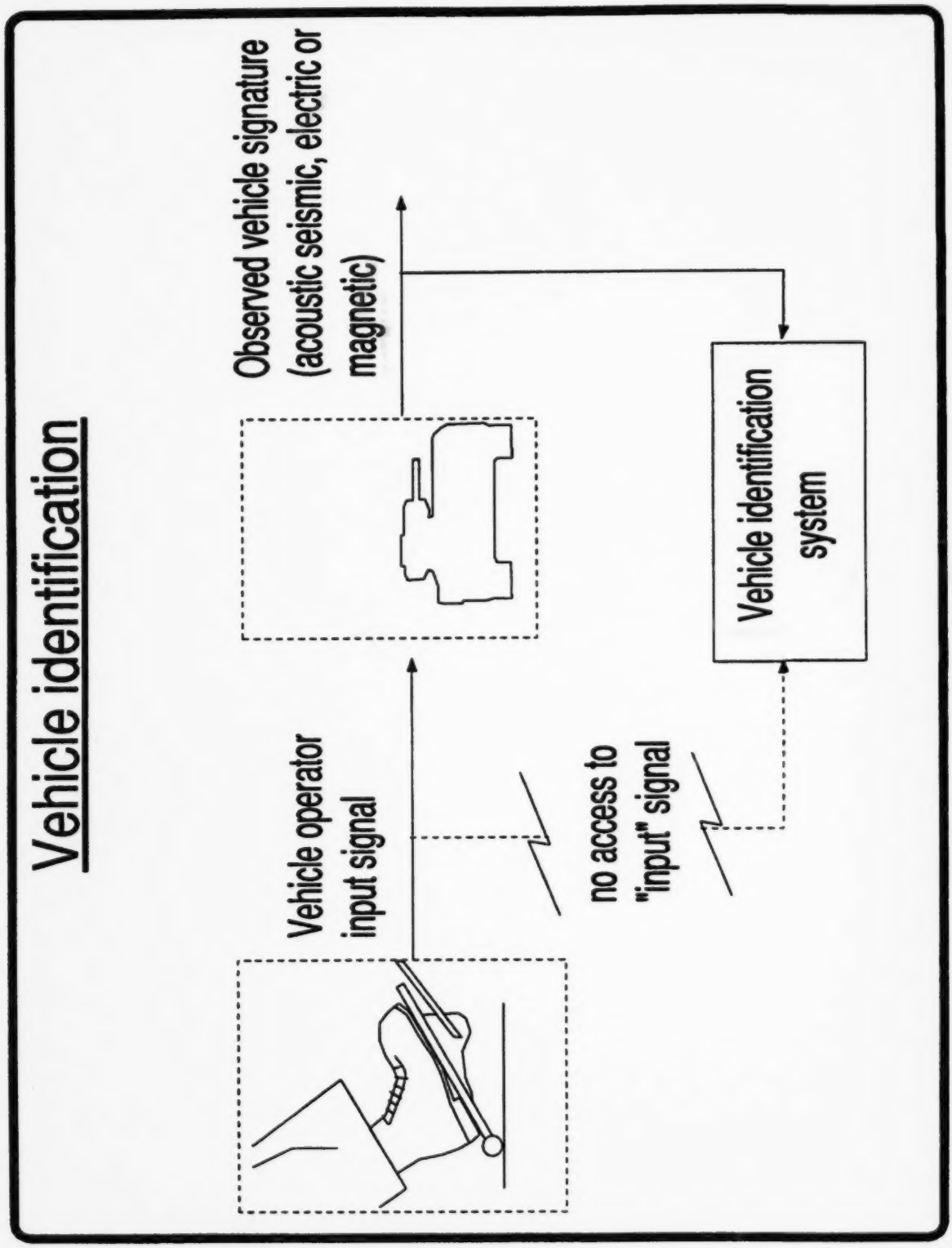
Classical system identification



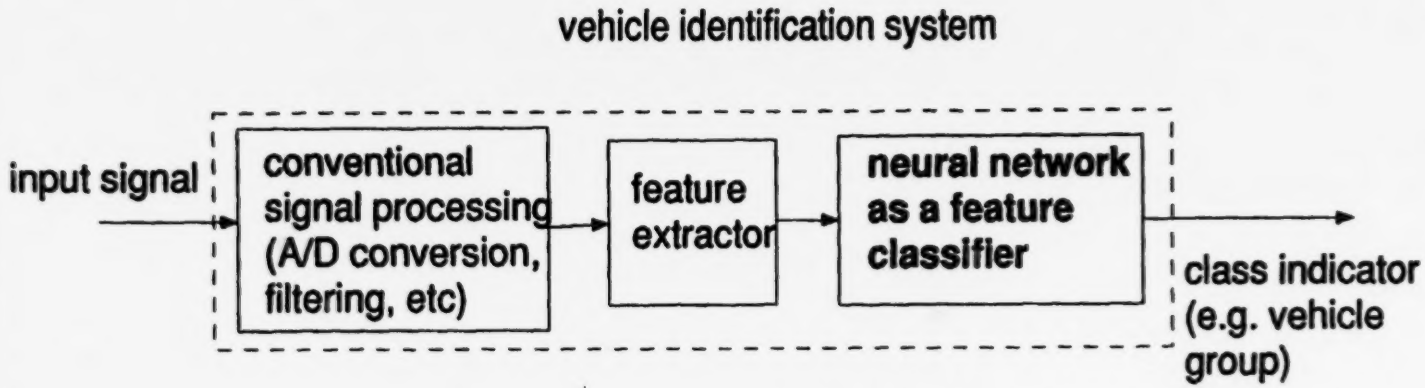
Plant process H is identified or approximated using only the observed input and output signals x(t) and y(t).

Applications:

adaptive systems(i.e. controller design for time-varying systems)


Vehicle identification
Observed vehicle signature (acoustic seismic, electric or magnetic)
Vehicle operator input signal
no access to "input" signal
Vehicle identification system

Artificial neural network application


vehicle identification system
input signal
conventional signal processing (A/D conversion, filtering, etc)
feature extractor
neural network as a feature classifier
class indicator (e.g. vehicle group)

Current Work:

1) Signal processing and feature selection/extraction

2) neural network simulations

Neural network simulations

Before attempting to use neural networks on the problem of vehicle identification, a practical expertise base will be built by first replicating some published results on the topic of neural networks.

The following simulations are among the projected applications of neural networks which have been started or will be simulated at DREV. This work is carried out in parallel with that of finding appropriate signal features enabling the discrimination of vehicles. The first two applications of neural networks are reported on in this presentation:

1) Application of the Adaline and the Least-Mean-Square (LMS) algorithm as a system identifier.

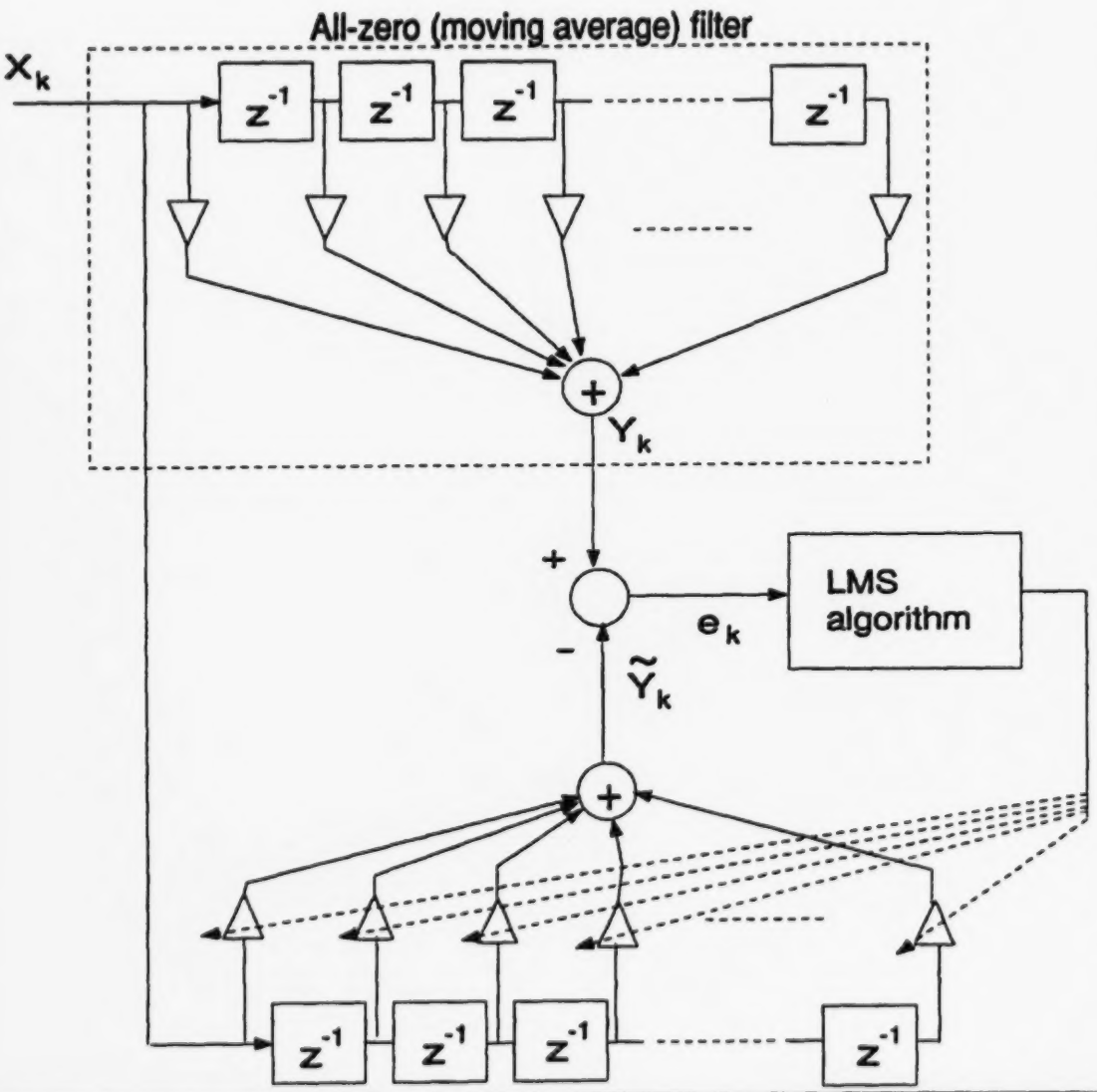
2) A layered neural network used for character recognition

3) A neural network used as an adaptive controller for an antenna array -- allows on-line control of the array's radiation pattern.

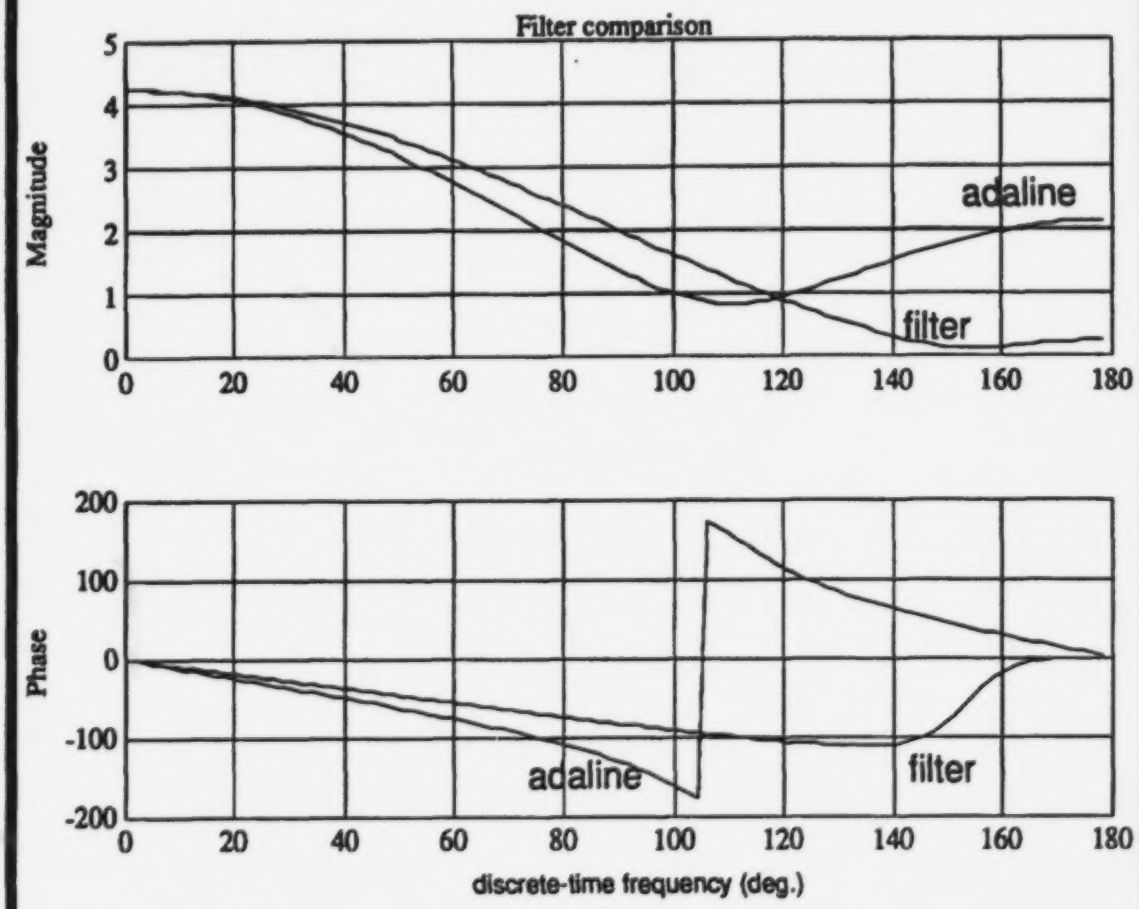
The ADALINE as a system identifier

Aim: Focus on the behaviour of the LMS algorithm when used to identify a known system.

LMS: $\vec{W}_{k+1} = \vec{W}_k + 2 \mu e_k \vec{X}_k$



Filter identification



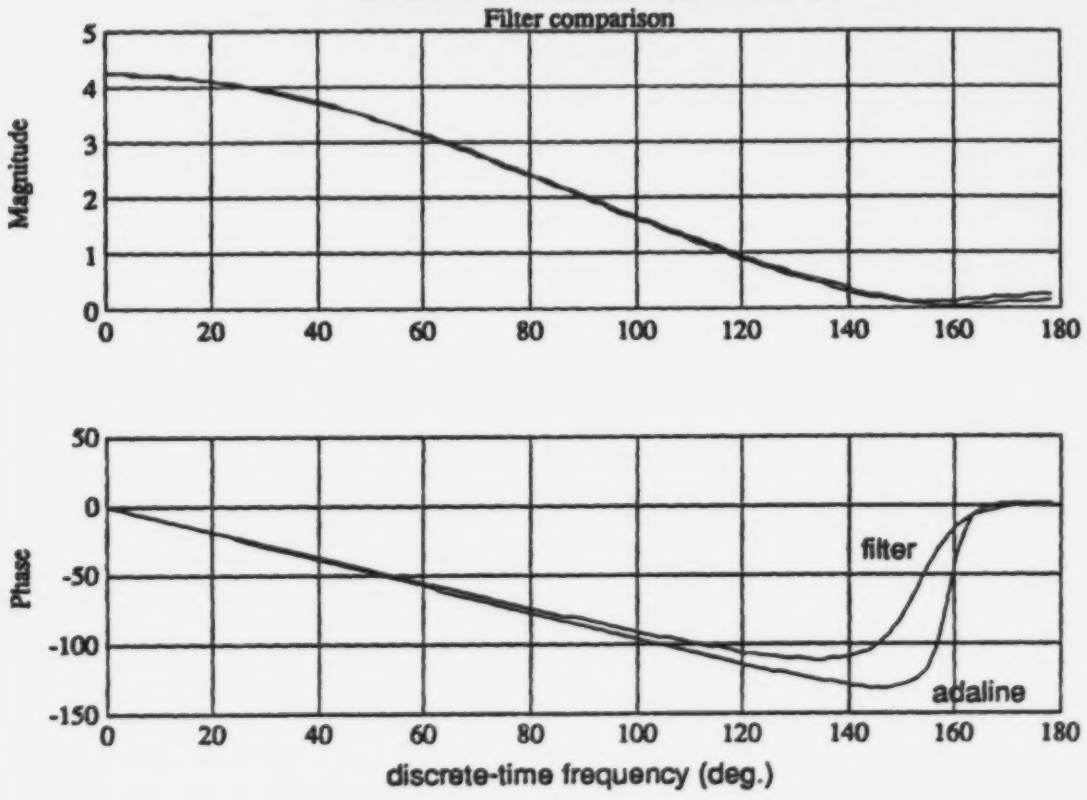
adaline order: 3

startup weight vector: 1, 1.9., 0.9

test signal: x = exp(-0.1 t)

mu=0.01

Filter identification



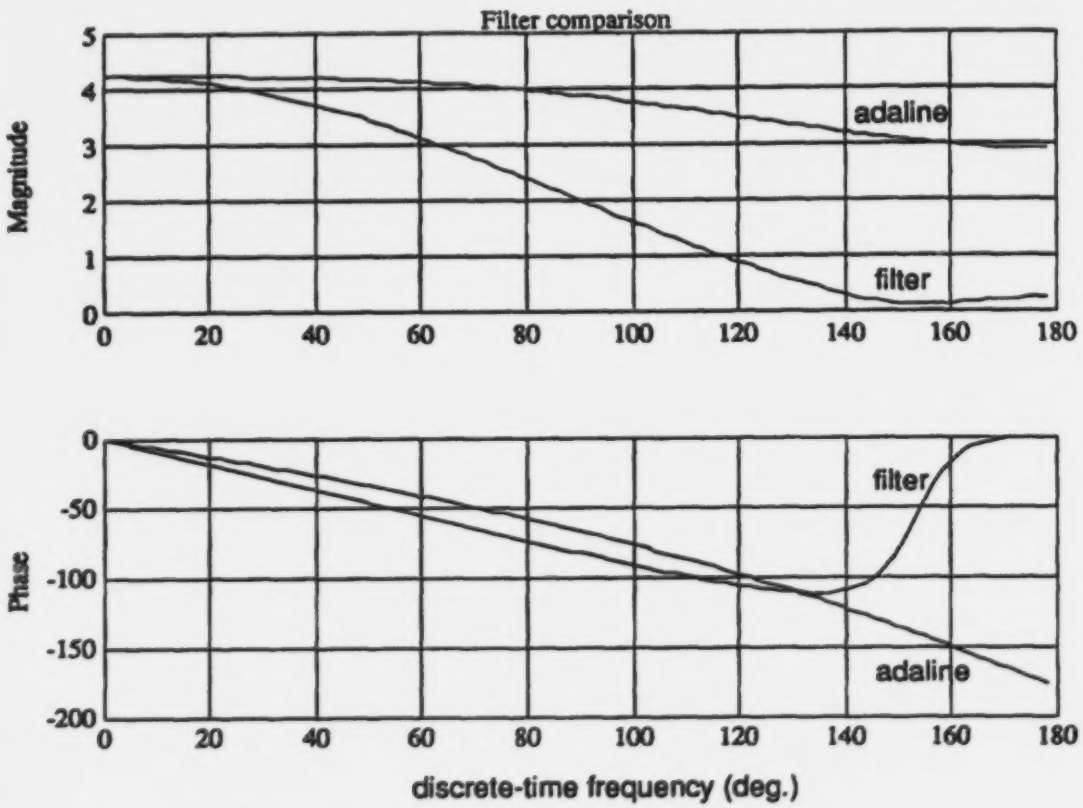
adaline order: 3

startup weight vector: 1, 1.9., 0.9

test signal: x = exp(-0.1 t)

mu=0.01

Filter identification



adaline order: 3

startup weight vector: .5, 3, -1

test signal: x = exp(-0.1 t)

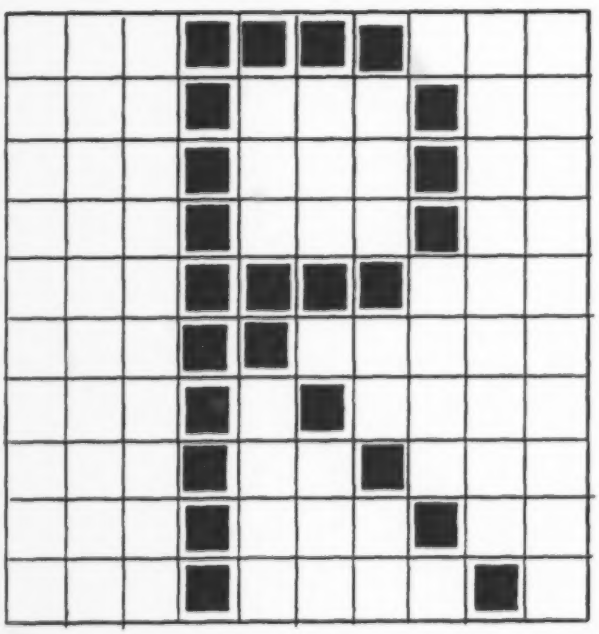
mu=0.01

Character Recognition with a neural network

Aim: Use the neural network as pattern classifier

Structure: 1) single adaline with sigmoid function

2) layered neural network

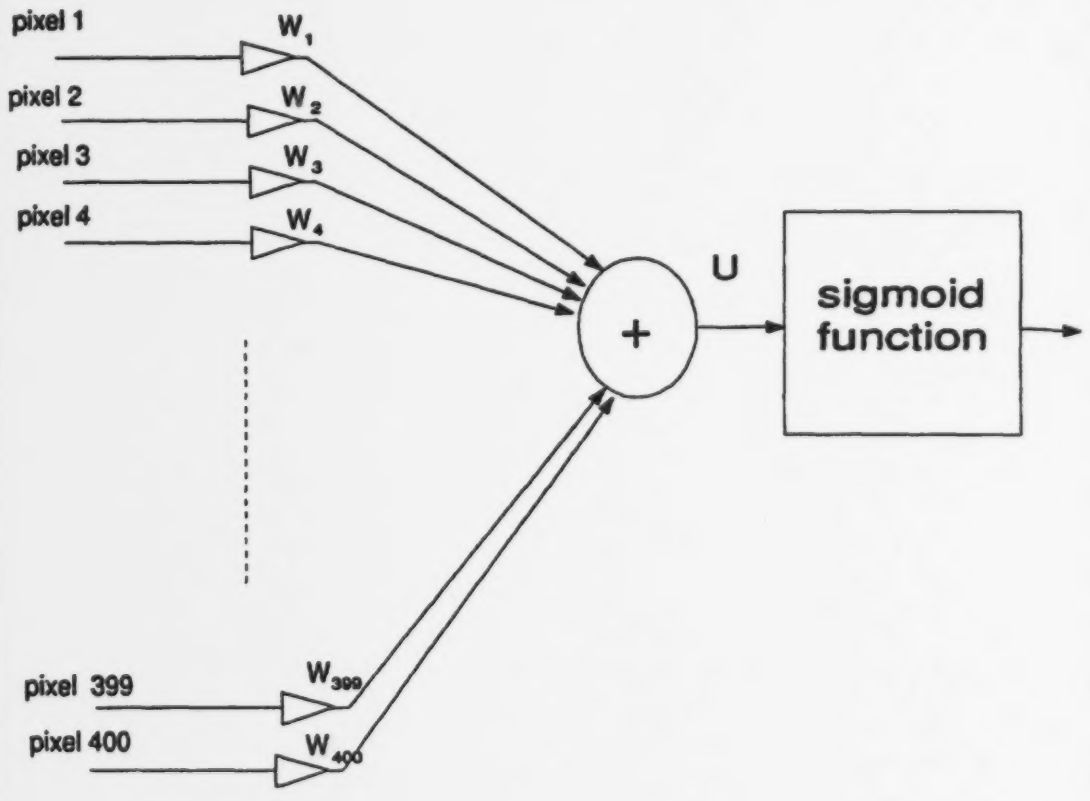


pixel values are binary:

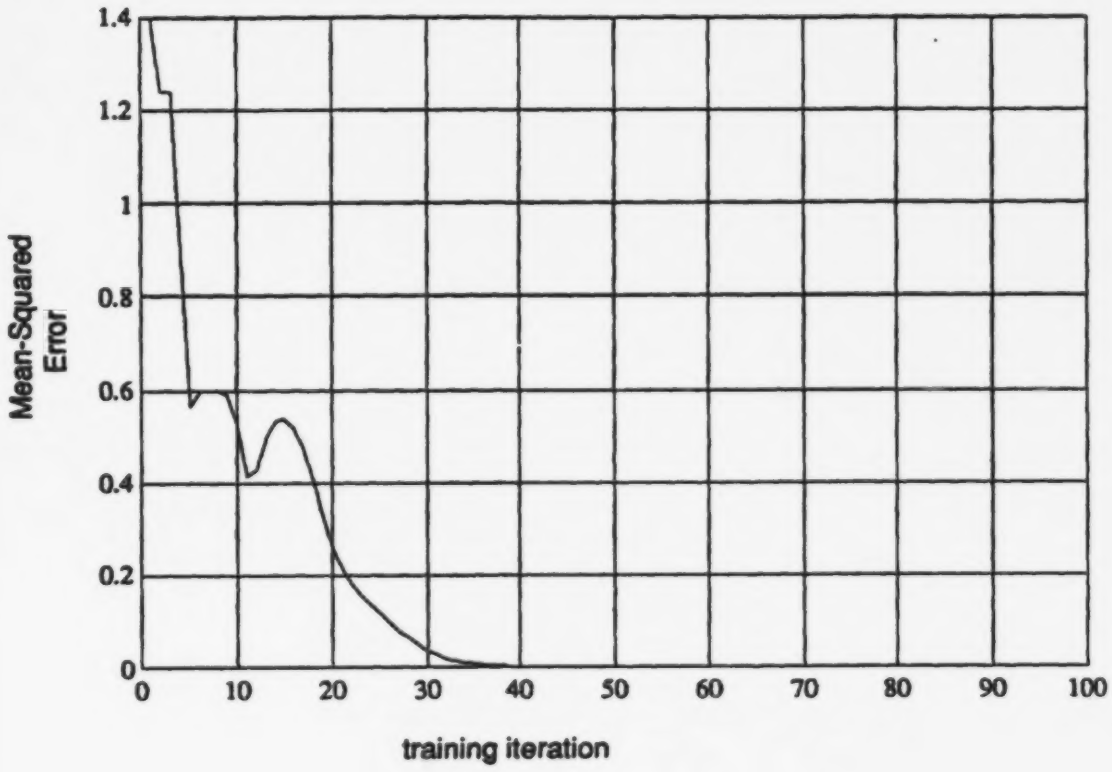
+1 and 0 or +1 and -1

Character Recognition with a neural network

1) single adaline:
   grid: 20 X 20
   algorithm: LMS
   sigmoid function: tanh(u) or 1/ (1 + exp(-u))



Character Recognition



Pattern Match

char #	target	adaline out	error(target-adaline
1 (C)	0.2000	0.1974	0.0026
2 (R)	0.4000	0.3799	0.0201
3 (D)	0.0000	-0.0000	0.0000
4 (V)	-0.2000	-0.1974	-0.0026
5 (E)	-0.4000	-0.3799	-0.0201

mu=0.001

MSE : 0.000164

sigmoid function: tanh(u)

This page intentionally left blank.

Periodic and Chaotic Signal Generation in a Recursive Artificial Neural Network

Simon A. Barton
Defence Research Establishment Suffield,
Box 4000, Medicine Hat, Alberta, T1A 8K6, Canada

Abstract

Feigenbaum showed that a large class of nonlinear functions may recursively generate values that exhibit periodicity. The periodic length doubles as the magnitude of some parameter in the function increases, and the system finally becomes aperiodic, or chaotic.

A recursively connected network of nonlinear processing nodes, such as those using the sigmoidal function, may therefore be expected to exhibit chaotic behaviour under some circumstances. Hopfield showed that some recursive artificial neural networks are stable; i.e. they settle to a constant output pattern.

We show that a simple recursive network of only 3 sigmoidal nodes can exhibit cyclical behaviour whose periodicity depends on the magnitude of a node response parameter, s. High values of s produce chaotic signals. From the numerically estimated values of s for which each period doubling occurs, we obtain a value of 4.669 for Feigenbaum's universal constant delta (4.669201..), thus confirming that this is a chaotic system. Chaotic and complex cyclical patterns can also arise in networks containing many nodes.

The behaviour of artificial neural networks of recursive nonlinear processing elements can be rapidly moved from stability, through complex periodicity, to chaotic activity and back again, by only slight and subtle changes in a parameter that controls the steepness of the nonlinear node response functions. The practical implications for artificial neural network applications, and the possible biological significance of this effect are discussed.

1 Introduction

Consider a nonlinear function of a variable x, whose shape is determined by a parameter λ; i.e. $f(x;\lambda)$. Feigenbaum (Ref. 1) showed that for a large class of such functions, values generated recursively exhibit a periodicity that doubles as the magnitude of λ increases past well-defined values. Beyond a critical value of λ, recursive values of f become aperiodic and are said to be chaotic.

Iterative or recursive signal generation may be written

$$x_{i+1} = f(x_i;\lambda) \tag{1}$$

Thus if $x_{i+1} = x_i$ the period is 1, and x_i is a fixed point of f. If $x_{i+2} = x_i$ the period is 2, if $x_{i+4} = x_i$ the period is 4, and so on. If λ_n is the value of λ for which the n-th periodicity of x first occurs, Feigenbaum showed that λ_n converges geometrically in n to λ_∞, for which the system is aperiodic. He defined

$$\delta_n = \frac{\lambda_{n+1} - \lambda_n}{\lambda_{n+2} - \lambda_{n+1}} \tag{2}$$

and found that δ_n approaches a universal constant

$$\lim_{n\to\infty} \delta_n = \delta = 4.6692.. \tag{3}$$

for *all* nonlinear systems that exhibit period doubling.

A recursively connected network of nonlinear processing nodes, such as one using a sigmoid transfer function, can therefore be expected to generate periodic and chaotic signals under certain circumstances.

Recursive artificial neural networks are often referred to as Hopfield nets because of the pioneering work done by J.J. Hopfield. His original model (Ref. 2) employed a completely connected set of simple linear threshold elements (LTEs) as nodes. The output of each node was either on or off (0 or 1), depending on whether or not the sum of the weighted input signals was above the node threshold. The signals passing between nodes were modified by a matrix of weights, which were iteratively adjusted to minimize a Liapunov function. Hopfield's later work (Ref. 3) employed sigmoidal nodes. The output of a sigmoidal node is a continuous, nonlinear, monotonically increasing function of the summed input. For both types of network, Hopfield showed that stable network states, giving a constant output, were possible.

Bruck (Ref. 4) proved that Hopfield nets with LTEs *always* converge to stable or cyclic states of period 2 or 4, if the matrix of connection weights

between the nodes is symmetric or antisymmetric. He termed these "the only interesting cases", but also showed that for an arbitrary weight matrix there exists a network of order n that has a cycle of period $2^{n/3}$. For example, with 30 LTE nodes there exists a network of period 1024. Bruck did not consider sigmoidal nodes.

The presence and role of periodicity and chaos in biological systems has been discussed by Freeman and coworkers (Refs. 5-7), and a simulation of biological neurons by Chay (Ref. 8) also showed the presence of complex oscillations and chaos. A mathematical proof of the existence of period-doubling to chaos in sigmoidal neurons was also given by Wang (Ref. 9). The generation of chaotic behaviour in neural networks has been widely discussed (Refs. 10-15).

In this work we show that a simple recursive network of only 3 sigmoidal nodes can exhibit cyclical behaviour, with period doubling observed from 1,2,.. 1024. The periodicity depends on the magnitude of a node response parameter s, which determines the steepness of the sigmoidal curves. For higher values of s, chaotic behaviour results. From the values of s at which each period doubling occurs, we obtain an estimate (4.669) of Feigenbaum's universal constant δ, showing that the process does indeed lead to a chaotic system.

Beyond the first chaotic region, there exist smaller regions of s, for which cyclic behaviour and period doubling reoccur. We have observed regions of periodicity 7, 14, 28, 56,.., also 9, 18, 36, 72.., and 8, 16, 32, 64.. For even higher values of s, for which the sigmoids become almost step functions (quasi-LTEs), cycles of unpredictable period occur (3, 6, 5, 13, 4..).

2 Recursive Networks

We consider networks that are composed of highly connected processing nodes, the output of each node n being a sigmoidal function f_n of the total input signal x_n. We define

$$f_n(x_n; s, x_{0n}) \equiv [1 + \exp(-s(x_n - x_{0n}))]^{-1} \tag{4}$$

The function f_n is parametric in a steepness factor s and an offset x_{0n}. Figure 1 shows the sigmoidal curves for $s = 4.0$ and 8.0, with an offset of 0.5. The offset plays the same role as a bias or threshold.

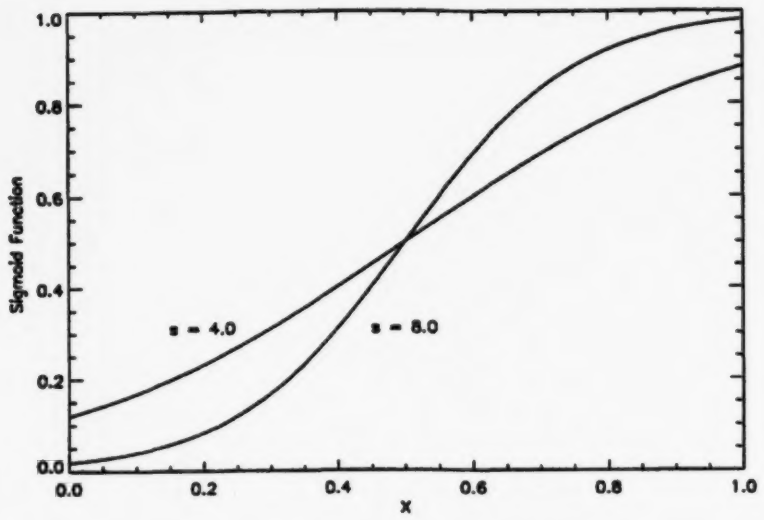


Figure 1: Sigmoid Functions with $s = 4.0$, 8.0, and $x_0 = 0.5$.

The total input signal to the n-th node at cycle t is

$$x_n^{(t)} = \sum_{m=1}^{N} W_{nm} f_m^{(t-1)} \tag{5}$$

i.e. it is the weighted sum of the outputs from the previous cycle of all the N nodes in the network. If the network is not completely connected, some of the weights will be zero. Also, the diagonal elements of the weight matrix $\mathbf{W}$ will be zero for nodes that do not have direct feedback.

In this work, we set $W_{nn} = 0$ for all n, and apply the following normalisation to the weights at each node (c is a constant)

$$\sum_m |W_{nm}| \equiv c \tag{6}$$

In principle, the steepness parameter s could be different for each node, but in this work the same value of s is applied to every node.

The recursive output of the network in matrix form is

$$\mathbf{f}^{(t)} = [1 + \exp(-s(\mathbf{W}\mathbf{f}^{(t-1)} - \mathbf{x}_0))]^{-1} \tag{7}$$

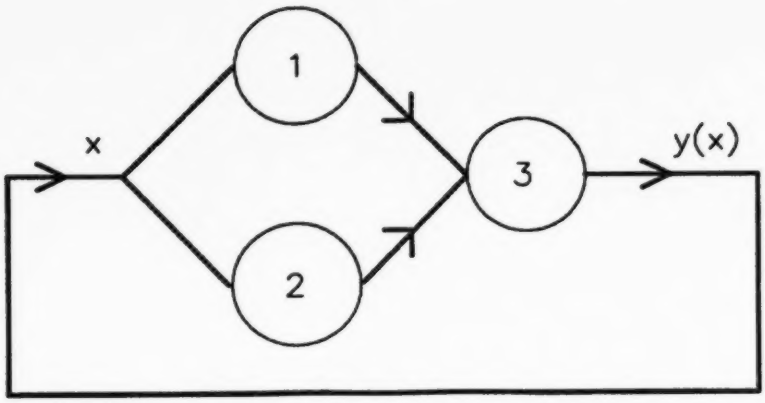


Figure 2: A Simple 3-Node Recursive Network.

The weights and offsets may themselves change from cycle to cycle. There are many weight updating schemes on the literature, most of which attempt to generate stable network states in response to given initial conditions; i.e. the vector **f** becomes fixed after a number of cycles. References 16 and 17 are collections that contain many of the important papers on weight updating algorithms. In this work, the weights are not altered from cycle to cycle.

In the next subsection we show how a simple 3-node system can be made to generate periodic and chaotic output.

2.1 A 3-node recursive network

Consider the simple recursive network of 3 nodes shown in Figure 2. Initially, we consider the function $y(x)$ that is generated by the group of 3 nodes in response to the input x. The output $y(x^{(t)})$ actually occurs after 2 network cycles; i.e.

$$y(x^{(t)}) = f_3^{(t+2)} \tag{8}$$

However, to demonstrate the property of period-doubling to chaos in a way that can be easily related to Feigenbaum's work (Ref. 1), we consider the iterative generation of the output of the 3-node group to be

$$x^{(t)} \rightarrow y(x^{(t)}) \rightarrow x^{(t+1)} \rightarrow y(x^{(t+1)}) \rightarrow \cdots \tag{9}$$

2.2 Generation of symmetric peak functions

The sigmoidal curves generated by nodes 1 and 2 of Figure 2 can be combined at node 3 in such a way that $y(x)$ is a symmetric peak function. This can be done by setting $W_{31} = -W_{32}$. Since nodes 1 and 2 are not connected, the weight matrix is

$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ -\frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \tag{10}$$

$\mathbf{W}$ satisfies the normalization condition (Eq. 6), and the node outputs are

$$f_n = [1 + \exp(-s(x - x_{0n}))]^{-1} \qquad (n = 1, 2) \tag{11}$$

and

$$f_3 = [1 + \exp(-s(p(x) - x_{03}))]^{-1} = y(x) \tag{12}$$

where

$$p(x) = \frac{1}{2}[f_2(x; s, x_{02}) - f_1(x; s, x_{01})] \tag{13}$$

If the two sigmoids f_1 and f_2 have different offsets, $p(x)$ is a peak function. Figure 3 shows $p(x)$ for several values of the steepness parameter (s = 10, 20, 30), with $x_{01} = 0.65$ and $x_{02} = 0.60$.

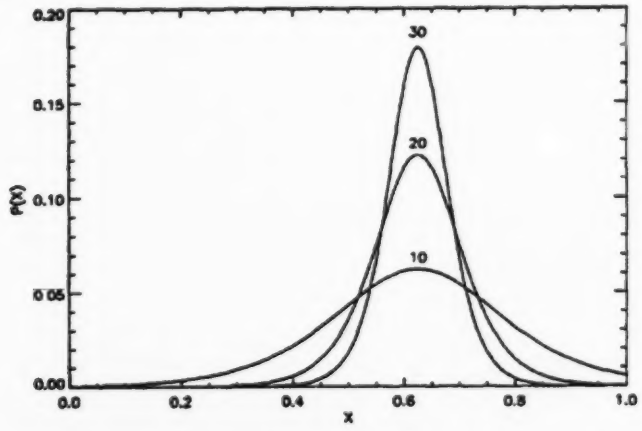


Figure 3: $p(x)$ with $s = 10, 20, 30$, and $x_{01} = 0.65$, $x_{02} = 0.60$.

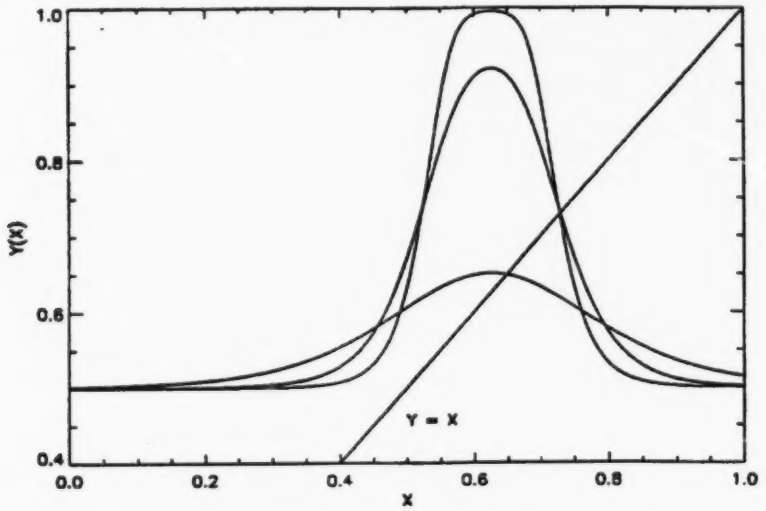


Figure 4: $y(x)$ with $s = 10$, 20, 30, and $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. The line $y = x$ is also shown.

The function $y(x)$ also depends parametrically on the offset for node 3, x_{03}. Figure 4 shows $y(x)$ with $x_{03} = 0.0$, and using the same parameters as those used in Figure 3. It can be shown that the asymptotic values of $y(x)$ are 0.5, and that as s increases the maximum approaches 1.0; i.e. for $x_{03} = 0.0$

$$0.5 \leq y(x) \leq 1.0 \qquad (-\infty < x < \infty) \qquad (14)$$

If x_{03} is not zero, the baseline of $y(x)$ moves up ($x_{03} < 0$) or down ($x_{03} > 0$) the y-axis.

2.3 Periodicity and chaos

Figure 4 shows the interception of the line $y = x$ with curves of $y(x; s)$ for 3 values of the steepness parameter s.

Feigenbaum showed (Ref. 1) that when the magnitude of the slope of $y(x)$ at $y = x$ is less than 1, the point of interception is a stable fixed point (x_f); i.e. iterations from all initial values of x converge to x_f. Thus, the lowest curve in Figure 4, with $s = 10$, will generate recursive values of about

0.65 for x and $y(x)$, for *any* initial value of x. This can be seen from the following procedure:

1. choose any initial value on the x-axis; e.g. $x_1 = 0.2$;
2. find $y(x_1)$ on the curve;
3. move horizontally to the line $y = x$ to find x_2;
4. iterate steps 2 and 3.

The iteration converges rapidly to the interception point on the curve with $s = 10$.

It is possible for a curve to have *more than one* stable fixed point. For example, consider the curve of Figure 5, which has been displaced to the right by setting $x_{01} = 0.8$ and $x_{02} = 0.75$, and has $s = 15.0$. In this case there are *two* stable fixed points, near 0.54 and 0.79.

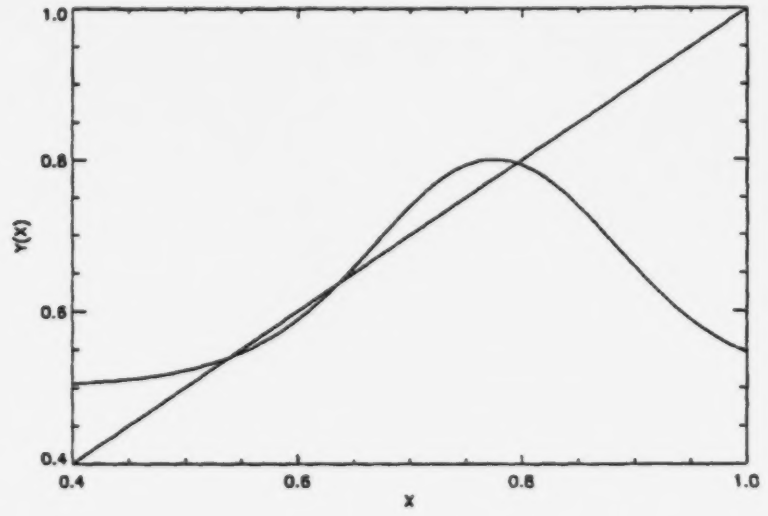


Figure 5: $y(x)$ with 2 stable fixed points; $s = 15$, $x_{01} = 0.8$, $x_{02} = 0.75$, $x_{03} = 0.0$. There is also an unstable point.

P131312.PDF [Page: 10 of 27

In Figure 5 there is also an *unstable* fixed point near 0.64. At this point the slope of $y(x)$ is > 1, and iterations diverge away from the point to either of the stable points. Therefore this system can stabilize to either of two values, depending on the initial value. There are certain regions of x for which the iterative result is extremely sensitive to the initial value; e.g. near 0.64 and 0.92 in Figure 5.

When many sigmoidal nodes provide input to another node, the output function can contain many peaks. Therefore, large Hopfield networks can stabilize to any of many fixed patterns, and which pattern results may be very sensitive to the initial conditions.

Consider the cases in Figure 4 where $|y'| > 1$ at $y = x$; i.e. with $s = 20$ and 30. Now the iterations diverge away from $y = x$. This is an unstable fixed point. As s increases from 10, there is a value for which the iteration settles into a cycle of period 2.

This *period doubling* occurs because the curve $y(y(x))$, which we write $y^{(2)}(x)$, has 2 stable fixed points. An example is shown in Figure 6, for $s = 15.0$.

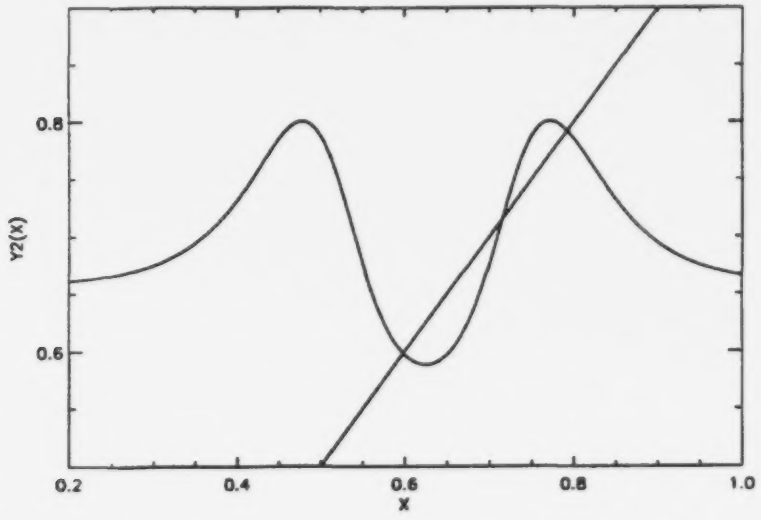


Figure 6: $y^{(2)}(x)$ with 2 stable fixed points; $s = 15$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. There is also an unstable point.

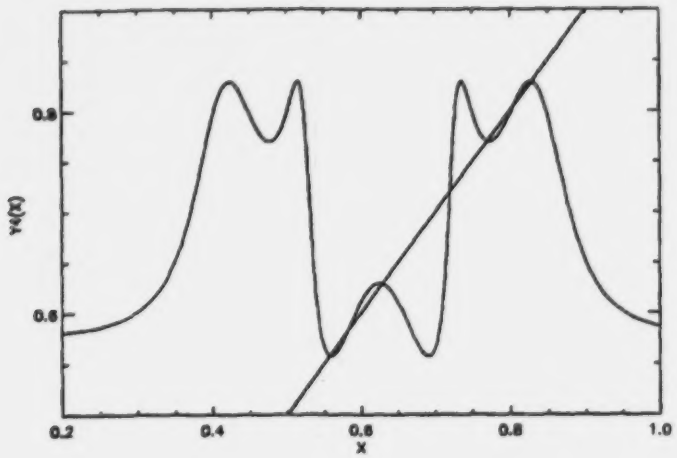


Figure 7: $y^{(4)}(x)$ with 4 stable fixed points; $s = 16$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. There are also 3 unstable points.

As s increases, the two stable points of $y^{(2)}$ become unstable, 4 new stable points first occur for $y^{(4)}$ (Figure 7, with $s = 16.0$), then 8 for $y^{(8)}$ (Figure 8, $s = 16.5$) and so on. Period doubling occurs *ad infinitum*, but the spacing between successive values of s at which doubling occurs becomes geometrically smaller. Thus there is a rapid convergence to a value s_∞, beyond which the system is aperiodic, or chaotic.

Table 1 lists the values of s_n, found numerically, at which the n-th period doubling was first observed to occur. Table 1 also gives the values of Feigenbaum's constants δ_n, calculated by substituting s_n for λ_n in Eq. 2. This shows a rapid convergence to Feigenbaum's value of δ (4.669201..).

It is quite difficult to obtain accurate numerical estimates of s_n, for several reasons. First, for any value of s that is very near a period doubling transition, it takes hundreds of thousands of iterations of the x-values to stabilize to a precise fixed cycle. For example, Figure 9 shows the slow convergence of iterated x-values when $s = 12.9$, near the first period doubling. As s approaches s_1 (12.9735..) the convergence gets slower, and as s increases further a stable oscillator of period 2 eventually emerges, as shown in Figure 10 for $s = 13.1$.

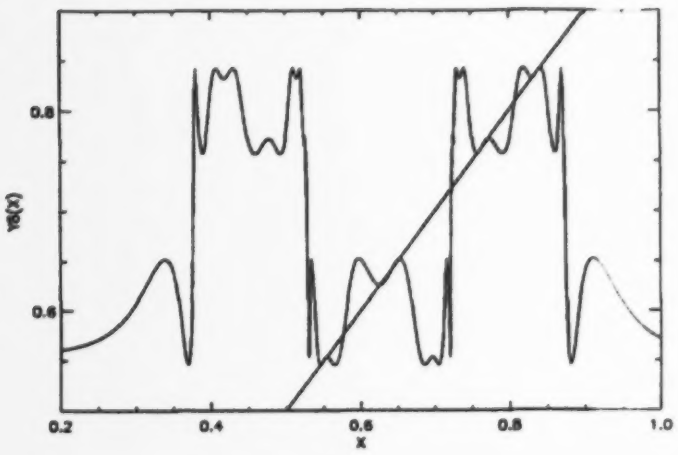


Figure 8: $y^{(8)}(x)$ **with 8 stable fixed points;** $s = 16.5$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$. **There are also 7 unstable points.**

n	s_n	period	δ_n
1	12.9735	2	2.503
2	15.4103	4	4.124
3	16.3838	8	4.579
4	16.61984	16	4.646
5	16.6713832	32	4.665
6	16.6824763	64	4.669
7	16.6848542	128	-
8	16.68536348	256	-

Table 1: Values of s **at which period doubling first occurs**

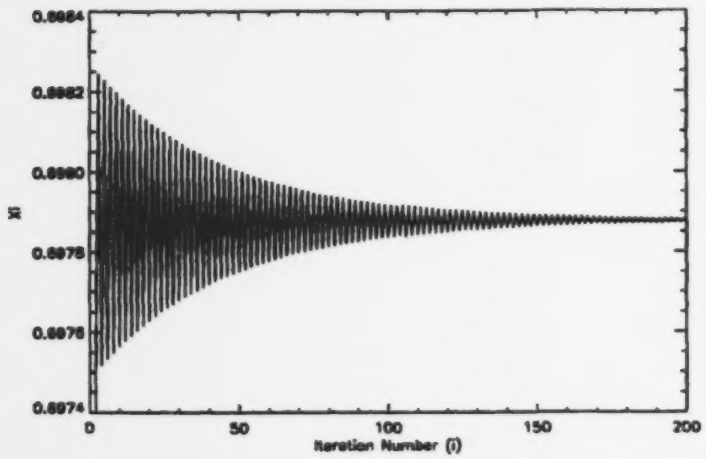


Figure 9: Slow convergence of x, when $s = 12.9$, near the first period doubling .

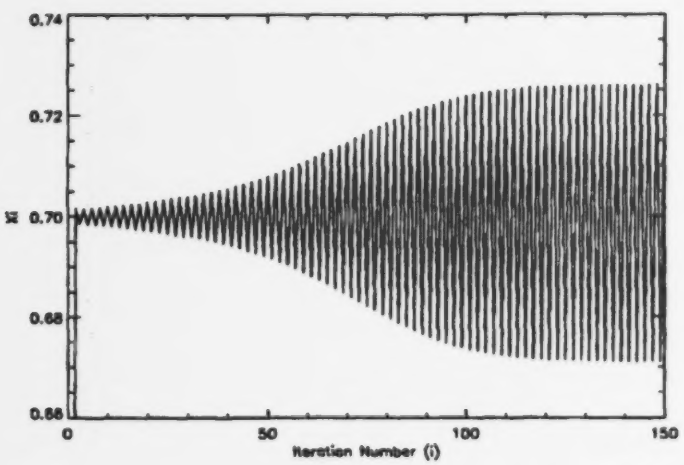


Figure 10: Convergence to a cycle of period 2 ($s = 13.1$).

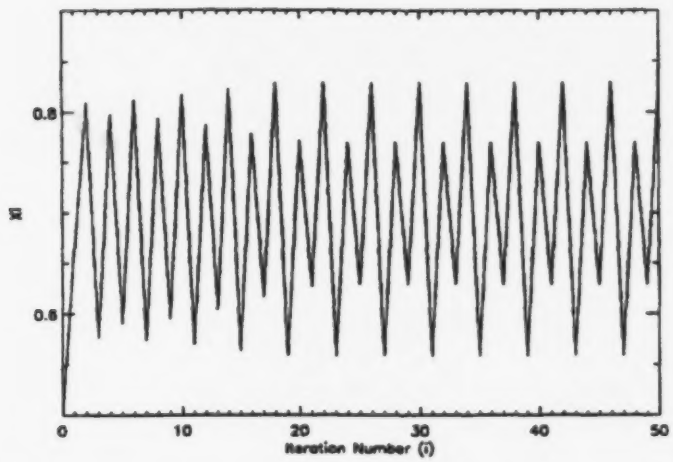


Figure 11: Cycle of period 4 ($s = 16.0$).

The closer s is to a period-doubling transition, the greater is the number of iterations required to determine the stable cycle, and so some criteria must be chosen to define the point at which period doubling is accepted to have occurred. Our algorithm performs the following two tests for a cycle of period N:

$$|x_i - x_{i-N}| < \Delta \tag{15}$$

and

$$|x_i - x_{i-M}| > \epsilon \qquad (M < N) \tag{16}$$

We used Δ = 1.0E-12 and ϵ = 1.0E-08. The second condition, Eq. 16, establishes the uniqueness of at least one other member of a cycle of period N. After every iteration, the new point must be checked against up to N previous points. Thus, if for example the period = 1024, thousands of checks must be performed on hundreds of thousands of iterations. This is quite CPU intensive, taking several minutes on a typical Unix workstation. We used a DECstation 5000/125. All calculations were done in double precision, using a C program.

Figures 11 and 12 show example of cycles of periods 4 and 8 respectively. They show that when the value of s is *not* near a transition, a stable cycle is quickly established after only a few iterations.

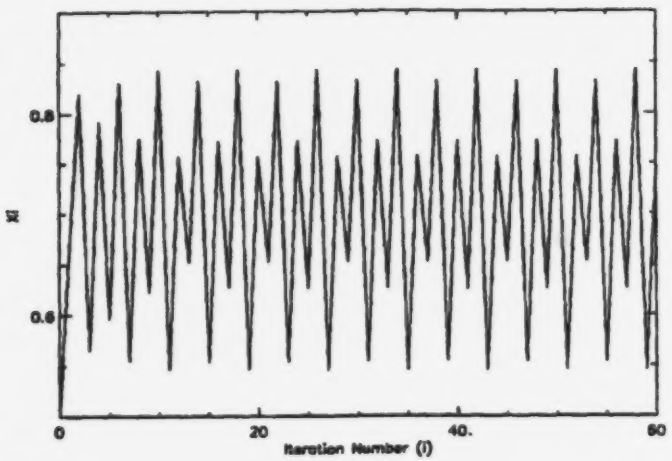


Figure 12: Cycle of period 8 ($s = 16.5$).

Table 1 shows values of s that generate period doublings up to 256. Cycles of period 512 and 1024 were also observed with $s = 16.6854726$ and 16.6854960 respectively, but no accurate estimates of s_9 and s_{10} were attempted numerically because the differences are approaching the limit of double precision accuracy.

Table 1 also indicates that a region of chaotic (aperiodic) signal generation would be expected for values of $s > 16.7$, and in fact no periodicity was observed immediately beyond this value. Figure 13 shows an example of chaotic iterative x values for $s = 17.0$. The values shown are an arbitrary set of 200 consecutive iterations.

Iterative periodicity and chaos can be represented as trajectories in xy-space. If we take each input signal x_i and its output y_i (i.e. x_{i+1}) to be a coordinate pair, then the movement of these points represents a trajectory. Figure 14 shows the trajectory for the cycle of period 8 that was shown in Figure 12. Figure 14 is actually a plot of 101 points from the 400th to 500th iterations. This shows that the system is constrained to move in a closed 8-point trajectory. Each of the 8 points must, of course, lie on the curve $y(x)$. Figure 15 is the trajectory for the chaotic system shown in Figure 13. In this case, the trajectory is *not* closed. Plotting more points results in completely filling the curve $y(x)$ to which the trajectory is confined.

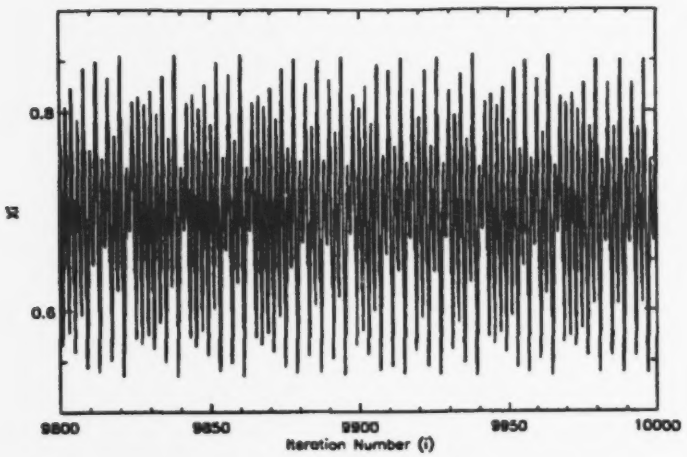


Figure 13: Chaotic signal generation ($s = 17.0$).

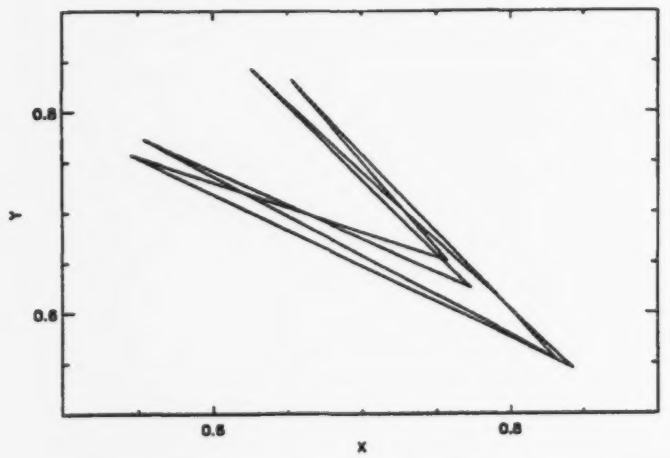


Figure 14: Trajectory of a cycle of period 8 ($s = 16.5$).

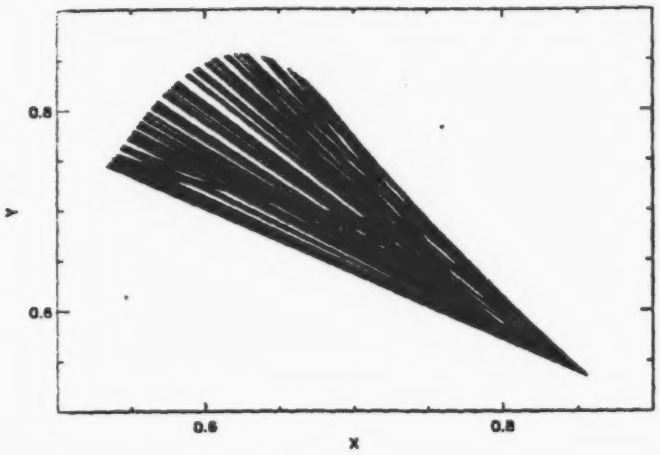


Figure 15: Chaotic trajectory ($s = 17.0$).

As s increases, the system remains chaotic until a cycle of period 7 appears near $s = 20.0$. Period doubling then reoccurs as s increases, as summarized in Table 2. The way in which this region of periodicity emerges from a chaotic region can be understood by looking at the curve $y^{(7)}(x)$ for $s = 20.0$, where the period is 7. Figure 16 shows that this curve has 7 points where $|y'| < 1$, where the line $y = x$ intercepts maxima or minima. Figure 17 shows the same curve, but with $s = 19.0$, just before the onset of periodicity.

n	s_n	period
0	19.996	7
1	20.090	14
2	20.131	28
3	20.140	56
4	20.1415	112
5	20.1419	224

Table 2: Approximate values of s in second period doubling region

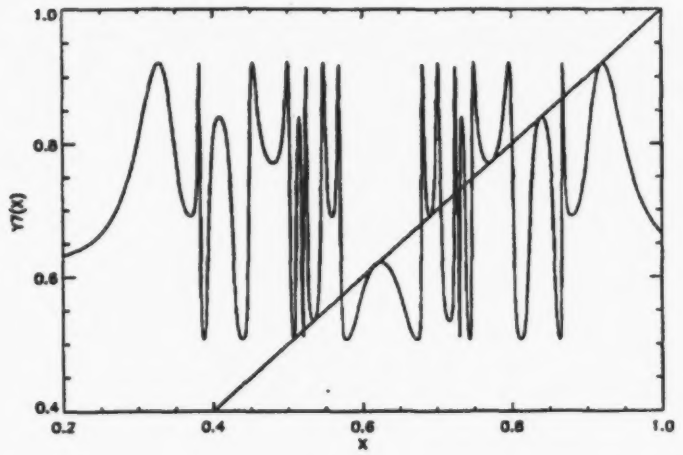


Figure 16: $y^{(7)}(x)$ with 7 stable fixed points; $s = 20.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

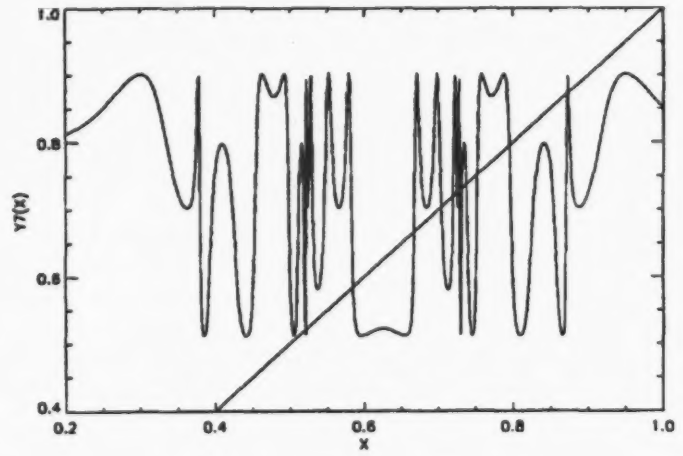


Figure 17: $y^{(7)}(x)$ with no stable fixed points; $s = 19.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

n	s_n	period
0	20.600	9
1	20.602	18
2	20.6032	36
3	20.60351	72
4	20.60358	144
0	20.95	8
1	21.00	16
2	21.024	32
3	21.029	64
4	21.030	128

Table 3: Approximate values of s in 3rd and 4th period doubling regions

The second region of periodicity spans a much smaller range of s than the first, and at $s = 20.15$ the system is again chaotic. However, because of the emergence of new maxima and minima in the curves $y^{(n)}(x)$, further regions of period doubling occur. At $s = 20.60$, a cycle of period 9 is established, period doubling reoccurs, then again at $s = 20.95$ a cycle of period 8 appears. These 3rd and 4th regions of period doubling are summarized in Table 3.

Beyond the 4th period doubling region the system is chaotic until about $s = 24$, where a cycle of period 3 appears. Now the sigmoids are so steep that they are almost step functions. At $s = 23.0$ (Figure 18) $y^{(3)}(x)$ has no value for which $|y'| < 1$ where $y = x$, but when $s = 25.0$ (Figure 19) there are three stable fixed points. This system of period 3 remains stable as s increases from 24 to 47; i.e. no period doubling occurs. This is due to the flat extrema produced by the quasi-step functions.

At $s = 48$, a cycle of period 6 occurs. However, this is not a process of period doubling leading to chaos, but rather a series of abrupt and unpredictable transitions. As s increases, cycles of period 5, 13, 4, 6, 5 and 7 appear for $s = 50$, 51, 52, 61, 62 and 70 respectively. Figures 20 and 21, showing $y^{(3)}(x)$ for $s = 47$ and 49 respectively, demonstrate the origin of these sharp transitions, arising near discontinuities in the steep quasi-step functions. Figure 22 shows the 6 fixed points for $s = 49$, and Figures 23 to 25 are examples of cycles with periods 5, 13 and 4. In this domain of s the fixed points jump erratically from one flat region of a y iterate to another.

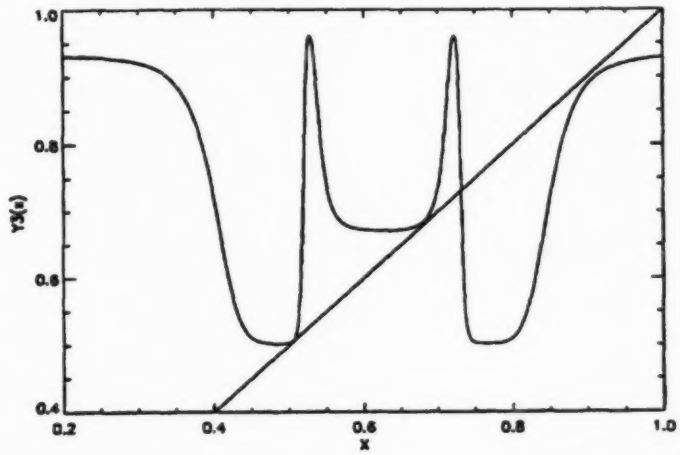


Figure 18: $y^{(3)}(x)$ with no stable fixed points; $s = 23.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

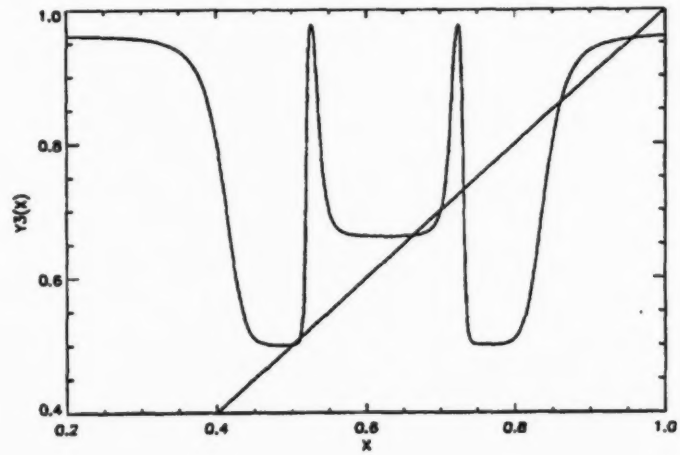


Figure 19: $y^{(3)}(x)$ with 3 stable fixed points; $s = 25.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

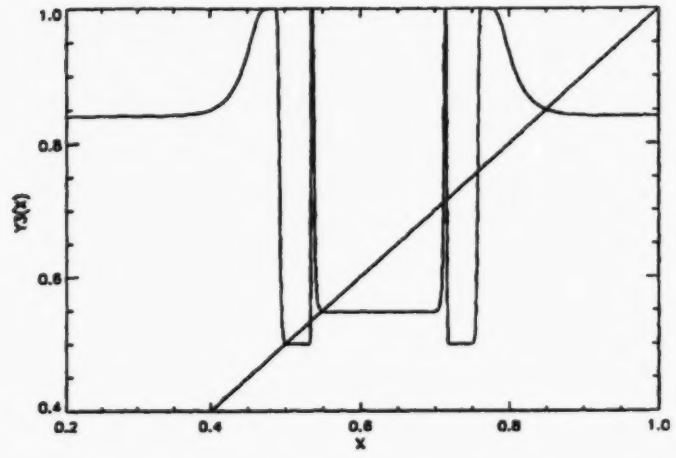


Figure 20: $y^{(3)}(x)$ **with 3 stable fixed points;** $s = 47.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

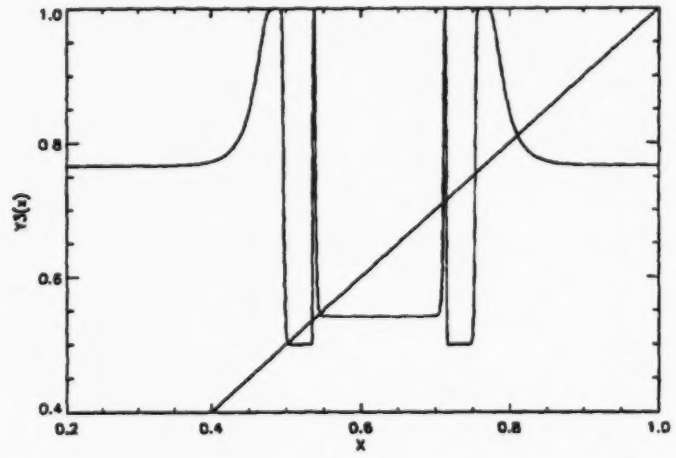


Figure 21: $y^{(3)}(x)$ **with no stable fixed points;** $s = 49.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

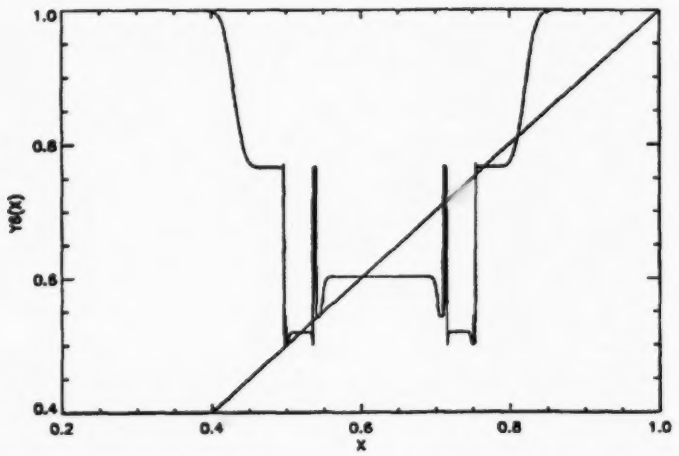


Figure 22: $y^{(6)}(x)$ with 6 stable fixed points; $s = 49.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

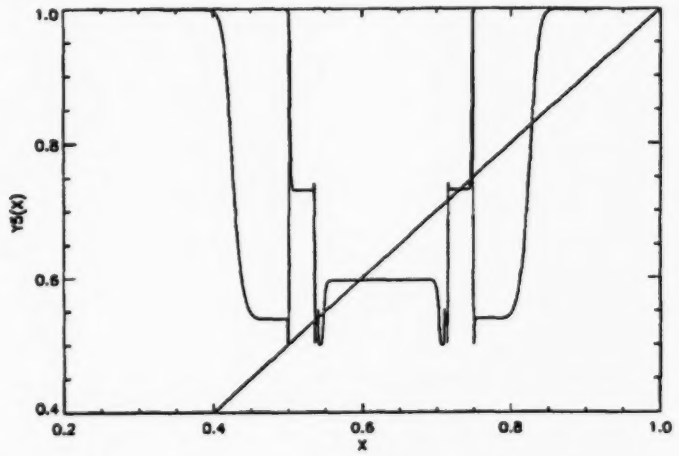


Figure 23: $y^{(5)}(x)$ with 5 stable fixed points; $s = 50.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

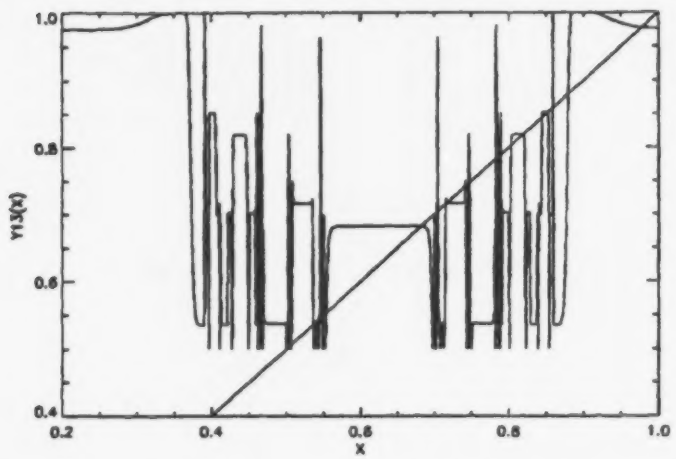


Figure 24: $y^{(13)}(x)$ with 13 stable fixed points; $s = 51.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

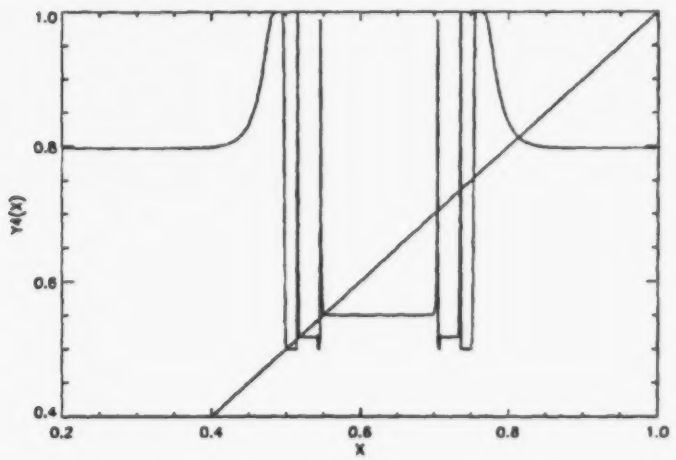


Figure 25: $y^{(4)}(x)$ with 4 stable fixed points; $s = 60.0$, $x_{01} = 0.65$, $x_{02} = 0.60$, $x_{03} = 0.0$.

2.4 Multinode recursive networks

We have simulated several completely connected multinode recursive networks. The nodes had no self-feedback ($W_{nn} = 0$). The weights for each node were chosen randomly between -1 and +1, and then normalized (Equation 6), with the normalization constant generally chosen to be the number of nodes in the network. The offsets, contained in the vector $\mathbf{x}_0$, were chosen randomly between 0 and 1.

The state at time t of a recursive network of N nodes is given by the vector $\mathbf{f}^{(t)}$. It is calculated from the state at the previous cycle using Equation 7. The initial components of the state vector for each network were chosen randomly between 0 and 1.

When a recursive network operates cyclically, periodicity must be detected by testing all the components of the state vector with all the components from previous cycles (cf. Equations 15 and 16). For example, for a 100 node network, the outputs from all the 100 nodes must be identical to those from a previous cycle for the system to be termed periodic.

From the work described in the previous section on 3-node networks, we expected that periodicity and chaos should occur for some random multinode networks, and that it should be possible to control this behaviour by adjusting the node steepness parameter s for the whole network.

Initial work on nets containing 20 and 100 nodes support this. A 20-node network showed cycles of period 1, 7, 14 as s increased from 1.0 to 2.1. However, beyond about 2.194, we could detect no periodicity. The 20-node system then appears to be chaotic. This aperiodicity persists until s reaches about 6.0, where a cycle of 1 occurs. For higher values of s, erratic periodicity occurs, much as in the 3-node case, with cycles of period 38, 1, 142, 1, 31, 33, 13,.. being observed in the region $s = 10$ to 20.

For the 100-node network, the system appears to be stable, i.e. converges to a cycle of period 1, for $s < 0.69$. Above this value we have detected no periodicity. The transition from stable cyclic behaviour to chaos seems to be much more abrupt for larger networks.

3 Conclusions

The iterative behaviour of a recursive network of sigmoidal nodes can converge to a stable or periodic state, or it may be chaotic. The behaviour of a 3-node system can be understood using the Feigenbaum model of a nonlinear system driven by single parameter, in this case the node response parameter s. As s increases, a stable system becomes periodic, and then exhibits period doubling to chaos. Beyond the first chaotic region, other regions of periodicity and chaos occur, and because of the step-like functions generated by higher s values, a region of erratic periodicity follows.

The behaviour of large recursive networks is much more difficult to analyse. However, it seems that the node response parameter s can be used to ensure stability. Below a certain value of s, a multi-node network will converge to a stable state. The onset of chaotic signal generation appears to be much more abrupt from our preliminary studies of networks of 20 and 100 nodes. The choice of the range for the weights and offsets, and the initial random network state vector, has a big effect on the behaviour of these larger systems. So far we have only studied a few cases. Much work remains to be done to understand more clearly the process of stability, periodicity and chaos in large recursive networks.

Depending on the exact initial state, a multinode net may converge iteratively to one of many possible stable states, and this has been proposed as a mechanism for pattern recognition. It should be possible to avoid iterative chaos in pattern recognition networks by adjusting the node steepness parameter.

On the other hand, there is strong evidence that chaotic signals are generated in at least some biological neural systems. In particular, Freeman *et al* (References 5-7) present evidence that the mammalian olfactory system generates chaotic signals during exhalation, i.e. with no input signal. Inhalation of a recognized odour appears to be associated with a cyclical periodic signal in the olfactory bulb. Chaotic signal generation may play a vital role in this neural system. Since the convergence to a particular pattern in a recursive neural system is critically dependent on the initial state of the system, chaos may provide an unbiased baseline upon which the incoming signal is imposed.

One can speculate on other uses of chaos in biological systems. If a system must respond in an unpredictable way, for example an insect such as a fly or mosquito that is disturbed by some movement, the generation of chaotic signals may assist its survival. Also, during the learning process, an

organism must generate many random movements in order for the neural system to learn the effects of a changing environment. In other words it learns by trial and error.

Some of the possibilities may be explored by constructing a simulated neural network that controls a simple model creature, and then allowing it to evolve in a simulated environment. The development of behaviour patterns may then be studied using different synaptic weight modification schemes. Our work in this area has the long-term goal of assisting the development of autonomous land vehicles, initially focussing on obstacle avoidance using input from an array of sensors.

4 References

1. Feigenbaum, M.J., "Universal Behaviour in Nonlinear Systems", Los Alamos Science, Summer 1980, pp. 4-27.

2. Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proc. Natl. Acad. Sci. USA, **79** (1982), pp. 2554-2558.

3. Hopfield, J.J., "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons", Proc. Natl. Acad. Sci. USA, **81** (1984), pp. 3088- 3092.

4. Bruck, J., "On the Convergence Properties of the Hopfield Model", Proc. IEEE, **78** (1990), pp. 1579-1585.

5. Skarda, C.A. and Freeman, W.J., "How Brains Make Chaos in Order to Make Sense of the World", Behavioural and Brain Sciences, **10** (1987), pp. 161-195.

6. Yao, Y. and Freeman, W.J., "Model of Biological Recognition with Spatially Chaotic Dynamics", Neural Networks, **3** (1990), pp. 153-170.

7. Freeman, W.J., "The Physiology of Perception", Scientific American, February 1991, pp. 78-85.

8. Chay, T.R., "Complex Oscillations and Chaos in a Simple Neuron Model", Proc. IJCNN, Vol. II, July 1991, pp. 657-662.

9. Wang, X, "Period Doublings to Chaos in a Simple Neural Network", Proc. IJCNN, Vol. II, July 1991, pp. 333-338.

10. Kurten, K.E. and Clark, J.W., "Chaos in Neural Systems", Physics Lett., **114A** (1986), pp. 413-418.

11. Kurten, K.E., "Critical Phenomena in Model Neural Networks", Physics Lett. A, **129** (1988), pp. 157-160.

12. Riedel, U., Kuhn, R. and van Hemmen, J.L., "Temporal Sequences and Chaos in Neural Nets", Phys. Rev. A, **38**, (1988), pp. 1105-1108.

13. Sompolinsky, H. and Crisanti, A., "Chaos in Random Neural Networks", Phys. Rev. Lett., **61** (1988), pp. 259-262.

14. Bauer, M. and Martiennssen, W., "Quasi-Periodicity Route to Chaos in Neural Networks", Europhys. Lett., **10** (1989), pp. 427-431.

15. van der Maas, H.L.J., Verschure, P.F.M.J. and Molenaar, P.C.M., "A Note on Chaotic Behaviour in Simple Neural Networks", Neural Networks, **3** (1990), pp. 119-122.

16. "Neurocomputing, Foundations of Research", Eds. Anderson J.A. and Rosenfeld, E., MIT Press, Cambridge MA, 1988.

17. "Neurocomputing 2, Directions for Research", Eds. Anderson J.A. and Rosenfeld, E., MIT Press, Cambridge MA, 1990.

Modular Neural Network Architectures

William C. Treurniet
Communications Research Centre

Abstract

The currently preferred approaches for automatic recognition of sounds in continuous speech hypothesize the presence of words or sub-words, such as syllables or phonemes, on the basis of statistical models derived from training data. This paper is concerned with the ability of artificial neural networks to model phonemes extracted from the DARPA TIMIT multi-speaker, continuous speech data base.

A number of proposed network architectures may be applied to the phoneme classification task. These range from the feedforward multilayer network and time delay neural network (TDNN) to more complex modular architectures which attempt to assign classifier modules to different regions of the input space. This paper describes experiments exploring the utility of some of these architectures including Hampshire and Waibel's Meta-Pi network, Nowlan and Hinton's system of experts, and Smieja's Minos architecture. In general, the results showed that these modular structures did not perform better than single networks trained with the same data. Some explanations for this lack of improvement are considered.

Two new modular architectures were also designed and tested. The first one employed 14 TDNN modules to map the complete input space to a different representation of reduced dimensionality. This reduced description was then used to train a multilayer network to discriminate among 38 classes. It was able to identify 52.4 percent of the test cases correctly. However, when the relatively high frequency *silence* class was ignored in generalization tests, the accuracy decreased to 45.8 percent.

A second modular system consisting of 49 TDNNs was also developed. Each network had one output unit and was trained to discriminate one phoneme class from all the others. In generalization tests, all modules received the same input data, and a classification decision was obtained by identifying the module with the largest output. This system of network modules achieved only 39.1 percent correct generalization.

The same data used to train the latter modular systems were used to train single multilayer networks and TDNNs. For 38 and 49 classes, the multilayer networks achieved 43.0 and 45.5 percent correct generalization, respectively, while the TDNNs' corresponding performances were 46.1 and 45.7 percent correct.

These experiments suggest that modular architectures do not produce improved generalization performance compared to single network structures. However, when computer memory is limited, a practical advantage may be gained by requiring fewer patterns at a time to train networks that each deal with a subset of the total number of classes.

1 Introduction

Automatic recognition of continuous speech from any speaker is a hard problem to solve by computers due to the variability in acoustic representations of utterances of the same word. However, the literature shows that a steady improvement in performance of laboratory systems has occurred since about 1975 [14]. Part of this improvement was due to better modelling of speech sounds [15], but further improvement has recently been shown using knowledge sources such as models of the task and users' goals to constrain the search space. For example, with no knowledge sources, one study showed 74 percent correct word recognition, but with the addition of knowledge sources such as the goals of the task, the recognition accuracy improved to 95 percent [29]. These reports show that speaker-independent recognition of continuous speech has been achieved and demonstrated to be useful for limited domain vocabularies.

Most commercially available speech recognizers typically are not "speaker-independent" in that they require training by the intended user. Further, the spoken words to be recognized must be clearly separated by gaps of silence to facilitate word end-point detection. These characteristics of speaker-dependence and discrete-word inputs severely limit the applicability of speech recognition technologies. Recent announcements of new products, such as the "Dragon-Dictate" speaker-adaptive recognizer from Dragon Systems, relaxes the speaker-dependence constraint somewhat by allowing the user to incrementally train the recognizer during use. However, this advance still does not provide the capability needed for conversational interactivity.

The limitations of current speech recognizers arise, to a large extent, from the template matching methodology employed. During a training phase, recognizers use discrete word utterances to form templates that represent the words in a vocabulary. Following training, an utterance to be recognized is compared to each stored template, and is identified as the word represented by the most similar template. Because this approach usually requires prior identification of word end points in the unknown speech, it is not easily extended to continuous speech recognition where word boundary detection is often inaccurate.[1] Further, speaker-independent recognition suffers since a template formed from multiple speakers' data usually results in poorer discrimination among vocabulary items than does one formed from a single speaker's data[2].

Since template matching methods are not particularly suited to speaker-independent, large vocabulary, continuous speech recognition, other methods have been devised and evaluated. In general, these attempt to deal with the variability among speakers and contexts by deriving probability distributions describing the likelihood of a speech unit given the evidence in the input data stream.

[1] But see [2] for a dynamic programming method that uses discrete-word templates to identify words in continuous speech.

[2] The severity of this problem may be reduced somewhat by forming multiple reference templates for a given word based on a cluster analysis of many utterances of the same word [28].

The most developed method of this type is the hidden Markov model (HMM) which is a finite state network where state transitions and symbol emissions depend on probability distributions derived from training data [20, 4]. More recently, a related approach has emerged based on neural network models trained to behave as classifiers. For a given input, such networks generate an output decision based on network parameters that implicitly model the probability distributions of features in training data [8]. A number of reports describe the classification behaviour of various neural network architectures based on the back-propagation learning algorithm [22]. These include the generic multilayer perceptron[16, 25] as well as the more sophisticated time-delay neural network [27, 26, 23].

This paper reviews the adaptation of the back-propagation algorithm to several existing architectures and objective functions, and describes experiments evaluating their performance in classifying phonemes extracted from the TIMIT multi-speaker, continuous speech data base. Two new architectures are also proposed and evaluated.

2 Implementation of the Back-Propagation Algorithm

The learning algorithm was implemented in C on a Mercury MC3200AT array processor installed on an IBM-PC compatible host computer running the MS-DOS operating system. Recently, the same software was installed on a Mercury MC860 processor hosted by a Sun Sparc2 workstation. The implementation was enhanced with some heuristics to accelerate learning. These included Fahlman's suggestions to add a constant of 0.1 to the derivative of the logistic to minimize the effect of the "flat spot" at either end of the function, and to increase very small output errors using a hyperbolic arctangent transform [6]. The learning rate adjustment heuristics known as the delta-bar-delta method [10] were implemented as well to speed gradient descent. When the delta-bar-delta heuristics were not used, the learning rate for connections entering a node was reduced in proportion to the fan-in (i.e., the number of incoming connections) at that node [19].

3 Speech Data Preparation

This section presents information regarding the speech data base and the methods employed to prepare the data. The TIMIT speech data base is briefly described, as well as the software developed to extract sets of balanced training and test data from the data base, and the pre-processing method adopted to prepare the data for presentation to the neural network models.

3.1 The TIMIT Data Base

The Texas Instruments (TI) and MIT laboratories collaborated with DARPA assistance to produce a corpus of continuous speech data spoken by multiple speakers. The result is called the DARPA TIMIT Acoustic Phonetic Continuous Speech Data Base. A prototype version made available by the National Institute of Standards [7], consists of speech from 420 male and female talkers from eight dialect regions, each speaking 10 different sentences.

The data for each sentence consists of three files; a 'txt' file contains the sentence in ASCII format, a 'adc' file contains a talker's digitized speech sampled at 16 KHz, and a 'phn' file contains a list of phonemes in the sentence accompanied by the starting and ending address of each phoneme in the 'adc' file. Thus, a phoneme may be extracted from the 'adc' file by first consulting the 'phn' file for its location. Table 1 presents a list of most of the phonemes in the data base, including common words containing the phonemes. Note the subtle differences in pronunciation among some of the phonemes.

The data base is valuable in that it enables speech laboratories to begin research on continuous speech recognition with a minimum of effort, and also enables comparisons of research results among laboratories when the same training and test data are used.

3.2 The Data Extraction Software

The strategy for training classifier networks with supervised learning algorithms requires a set of tokens which represents equally all the classes to be discriminated. It is important, therefore, that the training set be properly balanced with respect to variables such as sex, dialect, and sentence origin (i.e., TI vs MIT). It is also necessary to be able to exclude or include particular talkers in order to create a test set in which talkers are different from those in the training set.

To accomplish this, software was written which created a directory file for each phoneme. This file contained the following information about examples of the phoneme wherever they occurred in the data base: phoneme label, dialect, sex, speaker label, sentence label, 'adc' file start address, stop address, token length, left context phoneme label, left context token length, right context phoneme label, and right context token length.

A second program scanned the directory files to select phonemes according to specified criteria, obtained the digitized phoneme data from the appropriate 'adc' file, and created the spectrogram representation starting at the beginning of the left context phoneme and ending at the end of the right context phoneme. The above header information followed by the spectrogram data was appended to the data file for a particular phoneme class .

One data file for each phoneme was used for training networks to recognize that phoneme class. Two more data files for each phoneme were also created to test the generalization performance of trained networks. One set of phoneme data files (Test Set A) was used for

Table 1: Phonemes in the TIMIT Data Base

	Phoneme	Pronunciation		Phoneme	Pronunciation
Stops:			Vowels:		
	b	bee		iy	beet
	d	day		ih	bit
	g	gay		eh	bet
	p	pea		ey	bait
	t	tea		ae	bat
	k	key		aa	bott
	dx	muddy		aw	bout
	q	bat		ay	bite
Affricates:				ah	but
	jh	joke		ao	bought
	ch	choke		oy	boy
Fricatives:				ow	boat
	s	sea		uh	book
	sh	she		uw	boot
	z	zone		ux	toot
	zh	azure		er	bird
	f	fin		ax	about
	th	thin		ix	debit
	v	van		axr	butter
	dh	then		ax-h	suspect
Nasals:			Others:		
	m	mom		pau	pause
	n	noon		h#	non-speech
	ng	sing		cl	closures
	em	bottom			
	en	button			
	eng	washington			
	nx	winner			
Semivowels Glides					
	l	lay			
	r	ray			
	y	yacht			
	hh	hay			
	hv	ahead			
	el	bottle			

periodic testing during training to identify deleterious effects of over-learning. The other (Test Set B) was used to provide an independent assessment of generalization performance.

The training set and test sets were each obtained from a different group of speakers to permit speaker-independent evaluations of performance. The labels for speakers in Test Set A were *mklw0, mdwd0, madc0, fsjs0, mjls, fgmb0, fpad0, mrcs0, fgwr0,* and *fcmh1*, while those in Test Set B were *fsah0, fslb1, mjeb0, fgcs0, mjws0, fkdw0, fjsa0, mlih0, mjrk0, mses0,* and *mrlk0*. The remainder of the speakers made up the training set. The frequencies of phonemes in each set were distributed equally across sex and across the eight dialects. For consistency with others[15, 21], only data from the *si* and *sx* sentence types were included.

3.3 Speech Signal Processing

The time domain speech data sampled at 16 KHz was transformed to a frequency domain representation via a procedure described by Lang, Waibel, and Hinton [13]. A sequence of 320 points (20 msec) was pre-emphasized with a 6 dB/octave filter and processed by a Hamming window. Then the last 64 points from the 320 point window were folded into the first 64 points, resulting in a 256 point vector. These 256 values were processed by a 128 point complex fourier transform which treats even samples as real values and odd samples as imaginary. The resulting 128 values obtained from the FFT were transformed by the function $20 * \log_{10}(r^2 + i^2)$. The next frame of data was obtained by sliding the Hamming window forward in time by 64 points (4 msec) and repeating the process.

The output resulting from the above process was reduced to 16 values from 128 by forming weighted sums of the 128 values. That is, a 16 channel, third octave filter bank [1] was simulated by forming inner products of the output vector with each of 16 weights vectors[3]. The center frequencies of the simulated filters were 125, 185, 283, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, and 6300 Hz, forming an approximate mel scale filter bank. To reduce the amount of data even further, pairs of successive frames from the simulated filter bank were averaged to give a temporal resolution of eight msec.

4 The Multilayer Neural Network

The simple layered network structure with at least one hidden layer and with adjacent layers completely interconnected, has come to be called a multilayer network to distinguish it from the perceptron model which has no hidden layers. The perceptron is limited in that it can distinguish only sets of patterns that are linearly separable. The hidden layer(s) in the multilayer network removes this restriction by allowing hidden units to respond to

[3]The shape of each filter represented by a vector was triangular on a linear scale, with a bandwidth of .2316 times the center frequency, and 3 dB points at half the bandwidth from the center frequency on each side.

different combinations of input values which the output units can then combine to yield a non-linear segmentation of the input space. A schematic of such a "multilayer" network is presented in Figure 1.

5 The Time Delay Neural Network

The multilayer network typically represents the temporal dimension by mapping it onto a spatial input array. A consequence of this is that the network becomes very sensitive to the precise postioning of relevant features in the input array during training. That is, if a feature in different patterns of the same class appears at different positions in the input array, the effect is to smear the information presented to the network, and to reduce its sensitivity to the feature.

The Time Delay Neural Network (TDNN) structure, schematically presented in Figure 2, was specifically developed to circumvent this problem [12]. Each input node is presented sequentially with the value of a pattern element at several sucessive points in time. Since the same weights are adapted at each time step, they are trained to respond to the significant event wherever it occurs in the sequence. The outputs of the first layer of the network form the inputs to a second layer of weights. Here again, an input unit receives data from the successive time steps in the first layer, and the same weights (different from the first layer) are adapted at each time step. Finally, the outputs of the second layer's units are integrated across time to yield a final classification decision.

An important advantage of the TDNN structure, compared to the simpler multilayer network, is the stability of its outputs while a stream of data is passed through its input window. The TDNN will signal the presence of a feature as it moves into the window until it moves out again. The multilayer network, on the other hand, often responds erratically until the feature is properly aligned within the input window.

The dimensions of the TDNN structure employed here are similar to those recommended by Waibel et al. [26], and differed only in the network input width at a given delay due to the difference in frame rates. Their network input spanned 30 msec which corresponded to three spectrogram frames, while our network input spanned 32 msec or four frames. There were eight output nodes per delay at the first layer of the TDNN, and the input for the second processing layer spanned the result of five first layer time steps.

6 Experiments with Modular Architectures

An often cited difficulty with the back propagation algorithm applied to single structures such as the multilayer network and the TDNN, is that it does not scale up well because of cross-talk. Crosstalk occurs when a hidden unit receives conflicting error information from

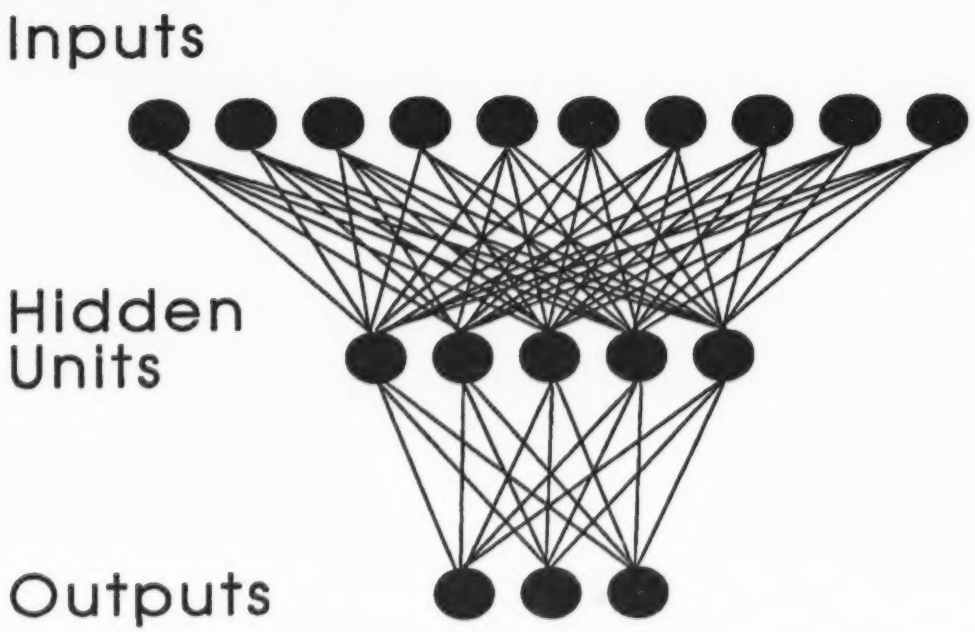


Figure 1: A multilayer network structure

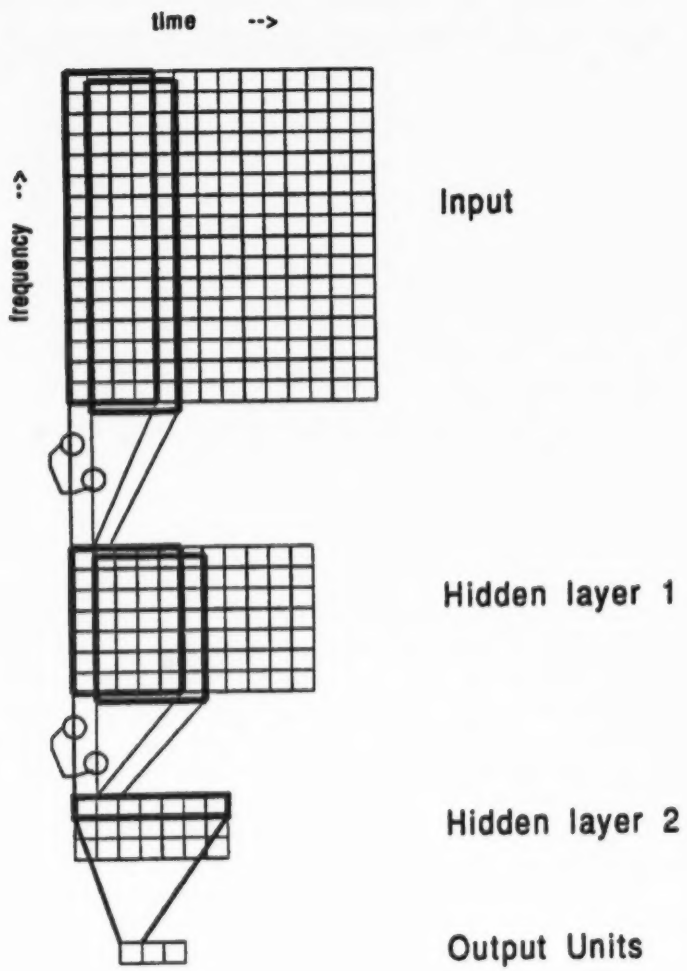


Figure 2: The Time Delay Neural Network structure

different output units [11], and larger networks provide more opportunities for this to occur. Another problem with large networks is that there are more parameters to adjust, and the number of training examples required for accurate adjustment increases in proportion to the number of parameters. Therefore, it would seem advantageous to have smaller network modules learn to handle parts of the problem, and then to combine the decisions of the individual modules.

However, simply observing the biggest output from all the modules trained with examples from disjoint regions of the input space, cannot reliably determine the decision of the system as a whole. By way of analogy, a function determined from a finite number of data points by regression analysis may be used to predict a value from a new point drawn from the same distribution as the original set. However, if the data point were drawn from a different distribution where the function should have had a different form, the predicted value would be quite incorrect. The function would extrapolate into a region where it did not apply. Similarly, a back propagation network which uses a "semi-linear" activation function such as the logistic, will also give an unreliable classification when presented with examples from a region of the input space not sampled during training. That is, the network will interpolate along the function learned, but may not extrapolate accurately into unexplored regions.

To address this problem, several network architectures have been proposed which include one or more modules for identifying the part of the input space to which an unknown sample belongs, as well as modules which make classification decisions. These system elements enable the final classification decision to be based on the network module which is deemed to have meaningful output. The "Meta-Pi", "Minos", and "system of experts" network architectures, discussed below, are examples of this type of modular architecture. Also, two new modular architectures are proposed which scale down the learning task, but do not require identification of a region of input space.

Network architectures were compared with regard to their ability to generalize to test data not in the training set. Training continued until generalization performance using the validation test set failed to improve or began to decline. Because the experiments were exploratory and because training times usually were extensive, the values of more than one variable often were allowed to differ between experiments. Therefore, comparisons among experiments should be made with caution. For example, no attempt was made to control for the total number of weights in the different network architectures, and often the sizes of the training sets differed due to limited training data or limited memory for storing the training set[4]. However, the experiments are considered adequate for discovering clear advantages or disadvantages in network architectures. Only representative experiments are reported.

Each training token consisted of 14 frames (112 msec) of spectrogram data (unless otherwise noted); these included 3 frames of the left context phoneme and 11 frames from the phoneme to be classified. Data from the right context phoneme were also included to complete the token when necessary. If there were insufficient data in the phoneme and its left and right context, the token was completed with zero values.

[4]The array processor contained six Mbytes of memory which limited the number of examples that could be used at one time for training the neural networks.

6.1 The Meta-Pi Network

Hampshire and Waibel [9] presented an architecture consisting of a number of networks each already trained to classify stimuli from a restricted region of the input space. They proposed an additional network, called a "Meta-Pi" network, to modulate the output of each sub-network to produce a global output. In the generic architecture shown in Figure 3, the trained networks are the "subnets", and the Meta-Pi network is referred to as an "input region identifier". To obtain a global output, the outputs of each subnet are multiplied by an output of the Meta-Pi network and same-class outputs are summed. The winning classification is determined from the global outputs using a winner-take-all strategy. The Meta-Pi network, in effect, learns which subnet or combination of subnets to use to classify a particular input sample. The approach assumes that at least one of the subnets correctly classifies the input, since the linear combining function which computes the global output cannot invert the outputs of the individual subnets.

The Meta-Pi learning algorithm was derived for the case where the subnets all attempt to make the same classification decision (the conjoint classification task), and for the case where the subnets are trained to make different classification decisions (the disjoint classification task).

For the conjoint task, the nth global output is the sum of the nth output (ρ_n) from each of the K sub-networks weighted by the kth output of the Meta-Pi module corresponding to that sub-network. That is,

$$O_n = \frac{1}{\mu} \sum_k \rho_{kn} M_k \tag{1}$$

where

$$\mu = \sum_k M_k$$

The error derivative for the kth output of the Meta-Pi module is defined as

$$\frac{\partial E}{\partial M_k} = \frac{\partial E}{\partial \vec{O}} \frac{\partial \vec{O}}{\partial M_k}$$

From Equation 1

$$\frac{\partial O_n}{\partial M_k} = \frac{1}{\mu}(\rho_{kn} - O_n)$$

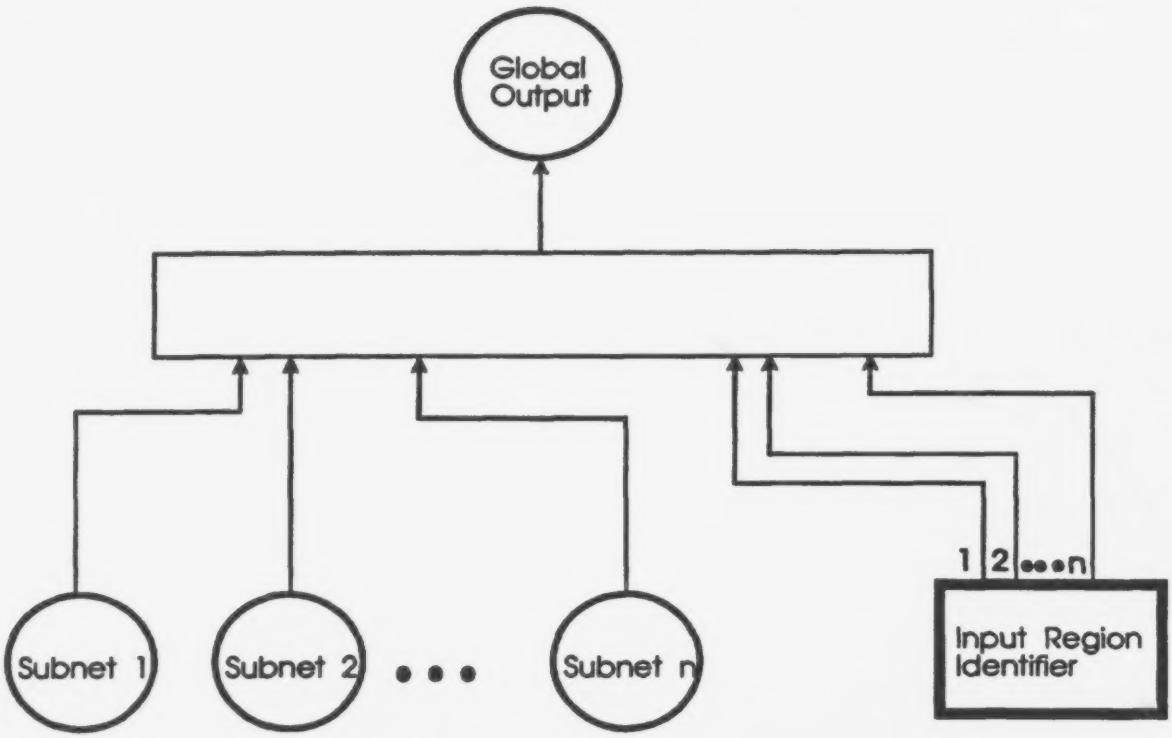


Figure 3: A generic modular architecture

If the error for the global output is defined as

$$E = \frac{1}{2}\sum_n (O_n - D_n)^2$$

where D_n is the desired output, then

$$\frac{\partial E}{\partial O_n} = O_n - D_n.$$

Therefore,

$$\frac{\partial E}{\partial M_k} = \frac{1}{\mu}\sum_n (O_n - D_n)(\rho_{kn} - O_n)$$

An example of the conjoint task was analysed by Hampshire and Waibel [9]. Each of six subnets was trained with data from a different speaker to discriminate among the phonemes *b*, *d*, and *g*. Then, the Meta-Pi network was trained to weight the three outputs of the six subnets so that the weighted sum of the subnet outputs correctly classified the sample. After training this architecture, a new sample could be classified without first identifying the speaker and the appropriate subnet. Instead, the Meta-Pi network took the responsibility of weighting the outputs of all the subnets on the basis of speech characteristics such as absolute position of the formants.

Hampshire and Waibel also proposed a variation of the learning algorithm for the case where each subnet is trained to make different rather than the same classifications. In this case, the above definition of the global output is modified to give k times n outputs instead of n outputs.

$$O_{kn} = \rho_{kn}\frac{M_k}{\mu} \tag{2}$$

From this definition

$$\frac{\partial O_{kn}}{\partial M_k} = \frac{1}{\mu}(\rho_{kn} - O_{kn})$$

and the error gradient, $\frac{\partial E}{\partial M_k}$, is modified accordingly.

Their proposal to test the algorithm was to train one subnet to discriminate among *b*, *d*, and *g*, and a second subnet to discriminate among *p*, *t*, and *k*. These sets of stop consonants differ in the degree of voicing present. Therefore, the Meta-Pi network would be expected to weight the subnet outputs in proportion to the amount of voicing present in the input sample. In this case, the number of global outputs is the total number of subnet outputs. Hampshire and Waibel [9] did not empirically investigate this form of the

Table 2: Meta-Pi Experiment with S, Z, R, and ER

Architecture	S	Z	R	ER	Overall
TDNN	81.0	74.1	85.8	69.1	79.1
TDNN	82.6	71.0			78.0
TDNN			89.3	74.5	84.5
Meta-Pi Net	82.2	69.8	82.2	74.5	78.3

Table 3: Meta-Pi Experiment with Voicing as the Cue

Architecture	P	T	K	B	D	G	Overall
TDNN	75.0	78.1	71.0				74.7
TDNN				80.2	69.0	76.5	73.4
Meta-Pi Net	36.6	40.8	26.6	61.3	39.4	49.0	40.2

Meta-Pi learning algorithm. Since combining disjoint classifiers is the problem of interest here, the performance of this form of the Meta-Pi algorithm was investigated. Several small experiments were performed with this architecture to weight the outputs of separate networks each trained to make different classification decisions.

In the first experiment, a single TDNN was trained to discriminate among the phonemes *s*, *z*, *r*, and *er*. Then, two more networks were trained to discriminate between *s* and *z*, and between *r* and *er*, respectively. The latter two networks became part of the Meta-Pi structure and its classification accuracy may be compared with that of the first TDNN. The results of the generalization tests, displayed in Table 2, show that the Meta-Pi architecture's performance was roughly equivalent to the TDNN network that learned to classify all four categories. Therefore, the Meta-Pi module, itself a TDNN network, successfully learned to distinguish the part of the input space containing *s* and *z* from the part containing *r* and *er*.

A potentially more difficult problem for the Meta-Pi network module was suggested by Hampshire and Waibel; that is, discriminating between voiced and unvoiced stop consonants. In this experiment, two TDNN networks were trained to discriminate among the voiced and unvoiced stop consonants, respectively. Then the outputs from these networks were weighted by the Meta-Pi module to produce a global output. The results of the generalization tests, displayed in Table 3, show a quite drastic decline in classification accuracy based on the global output as compared to the performance of the two separate TDNNs. Apparently, the regions of input space containing the voiced versus the unvoiced stop consonants were difficult to discriminate.

Another way to decompose the stop consonants is in terms of place of articulation. Three TDNNs were each trained to discriminate between two classes - *p* and *b*, *t* and *d*, and *k* and *g*, respectively. As before, a Meta-Pi module weighted the outputs of the three networks to produce a global output. Table 4 shows results of generalization tests on the three

Table 4: Meta-Pi Experiment with Place of Articulation as the Cue

Architecture	P	T	K	B	D	G	Overall
TDNN	80.4			87.7			83.9
TDNN		79.9			75.6		78.0
TDNN			82.2			78.4	81.8
Meta-Pi Net	49.1	65.7	49.7	70.8	52.0	52.9	56.9

TDNNs and on the Meta-Pi system. The individual TDNNs did well, but it was clear on closer examination that each discriminated primarily on the basis of the voicing cue. Consequently, the network trained with *p* and *b* would discriminate almost as well between *t* and *d*. The Meta-Pi module, which had learned to weight the other networks' outputs on the basis of place of articulation, did somewhat better than in the preceding experiment. However, performance was still considerably lower than that of the individual TDNNs.

6.2 The System of Experts

Nowlan and Hinton [18, 17] proposed a learning procedure for a system consisting of a number of separate networks, each of which learns to classify a subset of the training cases. At the same time, a "gating" network learns to weight the outputs of these "expert" networks, and causes the expert networks to compete with each other for control of particular regions of the input space. In the generic modular architecture shown in Figure 3, the expert networks are the "subnets" and the gating network is the "input region identifier". The gating network of a trained system may be considered a manager who knows which of the experts is best qualified to deal with the region of the input space from which a sample of interest is drawn.

The expert networks learn via a supervised learning algorithm which requires that the desired outputs be specified. However, the gating network's learning is unsupervised, and adaptation depends on the relative performances of the expert networks. The expert networks all receive the same input data which may, however, be different from the data presented to the gating network.

The model has been evaluated using a number of relatively small problems and performed as desired [17]. However, when a large number of classes are to be discriminated and the input space is large and complex, the model is computationally slow on current serial machines. This is because many candidate networks must be available during training (even though only a few may ultimately become experts), and because each candidate must have the structural potential for dealing with all of the classes. Nevertheless, because of its elegance, the model was considered an interesting candidate for decomposing a large task into smaller sub-tasks.

The objective function to be minimized for the system of *k* experts is the negative log

probability of generating the desired output vector under the mixture of Gaussians model employed [17].

$$L = -\log \sum_k (p_k e^{-\frac{1}{2}\|\mathbf{d}-\mathbf{y_k}\|^2/\sigma^2}) \tag{3}$$

where $\mathbf{d}$ is the desired output vector, $\mathbf{y}$ is the output of the kth expert network, p_k is the gate network output corresponding to the kth expert network, and

$$\| \mathbf{d} - \mathbf{y_k} \|^2 = \sum_i (d_i - y_{ki})^2$$

For convenience, let us define

$$L = -\log U$$

and

$$U = \sum_k (p_k e^{M_k})$$

where, for the kth expert net,

$$M_k = -\frac{1}{2} \sum_i (d_i - y_{ki})^2/\sigma^2$$

Then the error gradient for the ith output unit of the kth expert with respect to its input is

$$\frac{\partial L}{\partial x_{ki}} = \frac{\partial L}{\partial U} \frac{\partial U}{\partial M_k} \frac{\partial M_k}{\partial y_{ki}} \frac{\partial y_{ki}}{\partial x_{ki}}$$

Since

$$\frac{\partial L}{\partial U} = -\frac{1}{U}$$

$$\frac{\partial U}{\partial M_k} = p_k e^{M_k}$$

$$\frac{\partial M_k}{\partial y_{ki}} = (d_i - y_{ki})/\sigma^2$$

and, for the logistic activation function

$$y_j = \frac{1}{1 + e^{-x_j}}$$

$$\frac{\partial y_{ki}}{\partial x_{ki}} = y_{ki}(1 - y_{ki})$$

then

$$\frac{\partial L}{\partial x_{ki}} = -\frac{1}{U} p_k e^{M_k} (d_i - y_{ki}) y_{ki} (1 - y_{ki}) / \sigma^2$$

The error gradient for the ith gating network output is derived after first choosing the "softmax" activation function for the output units; that is, where x_i is the input to the ith output unit,

$$p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

The objective function then becomes

$$L = -\log \sum_i \left(\frac{e^{x_i}}{\sum_j e^{x_j}}\right) e^{M_i}$$

After some manipulations, keeping in mind that the index j includes i in the above softmax definition[5],

$$\frac{\partial L}{\partial x_i} = -\frac{1}{U} p_i e^{M_i} + p_i$$

The error gradients with respect to the inputs for the gating network and all expert networks are now propagated back through the networks as before to derive the error gradients with respect to all the weights.

At the beginning of learning, the gating network should consider all experts equally probable for any input sample. This is obtained by initializing to zero all the weights leading to the output units.

A system of experts architecture with 10 multilayer "expert" networks and one multilayer gating network was trained to discriminate the *s* and *z* phonemes. All networks had 240 inputs and 10 hidden nodes. The expert networks each had two outputs while the gating network had 10 outputs corresponding to the number of expert networks. The training set consisted of 200 samples per class.

Best generalization performance of the system was obtained after 1500 passes through the training set with *s* (81.0%) and *z* (72.2%). Overall performance was 77.5 percent correct. This result is very similar to that previously obtained with a single TDNN network.

A similar experiment was performed to discriminate among *w*, *l*, and *ow*. After 5550 training epochs, generalization performance, which was still improving very slowly, was *w* (90.9%), *l* (56.0%), and *ow* (75.0%). This is not as good as *w* (91.7%), *l* (67.3%), and *ow* (80.0%) obtained with a single multilayer network.

[5]The assistance of D. Graham, Canadian Space Agency, in confirming this equation is appreciated.

Clearly, the system of experts did not improve performance levels. Closer examination of several training runs revealed that each expert network selected by the gating network had learned to turn on only one of its outputs for any input sample. Therefore, the gating network was making all the decisions, and the system's performance depended only on the relative values of the gating network's outputs. Since only one network was doing all the work, the approximate equivalence of performance with that of a single network is not surprising.

The reason for the lack of improvement with the system of experts is not clear. One possibility is that the decision boundaries required were adequately represented already by a single multilayer network, so the task might have been too easy. It is also possible that any advantage from using the system of experts architecture might be demonstrated more easily if clearly disjoint regions of the input space are to be placed in the same class, as was the case in the vowel classification experiments reported by Nowlan [17]. The distribution of classes in the present experiments is difficult to determine due to the high dimensionality of the input space.

Another possibility is that the amount of training data, although adequate for a single network, was simply insufficient for training the several networks that eventually learned to specialize. Insufficient data might have led to grossly imbalanced class frequencies in a given expert's region of input space. Large differences in prior probabilities for the classes would then have adversely biased the network's learning, perhaps to the point of settling in a local minimum which only allowed one output to become activated. Increasing the training set size would help to reduce such an effect.

If a larger sample size is needed to enable the system of experts to perform properly, the architecture may be considered impractical since currently available computing equipment would take too long to handle all phoneme classes.

6.3 The Minos Architecture

The Minos architecture proposed by Smieja [24] is another approach for learning the region of input space that is relevant to a particular network module. In this sense, it is similar to the system of experts described above, but the learning rules are somewhat less elegant. It consists of a number of Minos modules which, for our task decomposition problem, each receive the same input data. Each Minos module consists of a worker network and a monitor network (Figure 4). The worker network attempts to learn to classify the input sample, and the monitor network learns how much confidence to have in the worker's decision about that particular input.

When learning to classify a given data sample, outputs are derived for both the worker and monitor of each Minos module. The performance of each worker is then assessed by comparing the similarity of the pattern of its outputs with the target pattern. Smieja recommends normalizing the output and target values so that the respective minima and

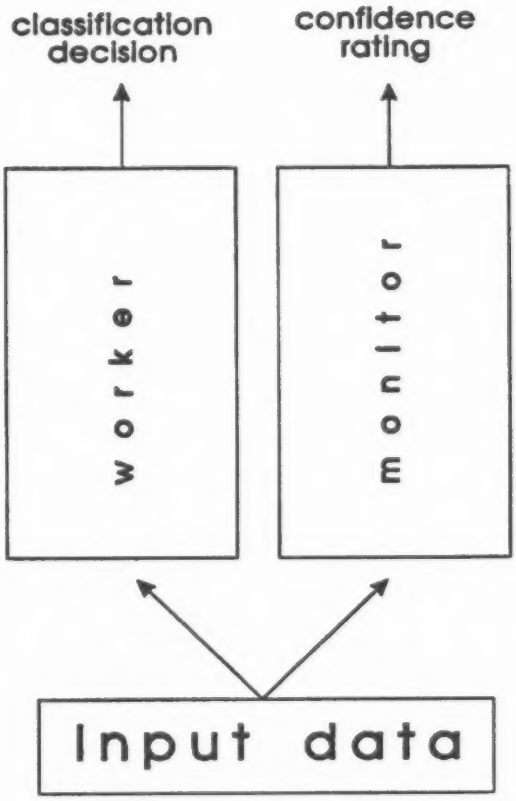


Figure 4: A Minos module

maxima are equal, then measuring the similarity of output and target with the absolute value metric. The worker with the highest similarity score is chosen to learn the pattern, while all other workers refrain from learning. However, all of the monitors learn. The winning worker's monitor learns to raise its confidence in its partner, while each losing worker's monitor learns to lower its confidence in its partner's performance for that input sample. In this way, a Minos module learns to specialize on a particular region and to communicate that a sample belongs to that region.

To classify an unknown sample after training, outputs are generated by all the monitors, as well as by the worker belonging to the monitor expressing the highest confidence. The final classification decision is obtained from this worker module.

A Minos architecture with 10 modules was trained to discriminate among *w*, *l*, and *ow*. Each worker network had 224 input units, 10 hidden units and three output units. Each of the monitor networks had the same number of input and hidden units, but had only one output unit. A monitor's target during learning was 1.0 to indicate high confidence, and 0.0 for low confidence. After 3200 epochs, the obtained generalization performance was *w* (81.3%), *l* (61.8%), and *ow* (76.7%). Overall performance of 69.1 percent correct was similar to that of the system of experts and about six percent less than that of a single multilayer network. Unlike the system of experts, however, the Minos workers did not exhibit an apparent local minimum that allowed only one output to turn on.

Again, there was no improvement in generalization performance over the single network. Performance may have improved with a larger training set since parts of it were eventually allocated to different Minos modules as they learned to specialize. However, as with the system of experts, a much larger training set size would have produced an unacceptable increase in training time.

6.3.1 Extension to the Minos Architecture

With the procedure described above, a worker module makes weight adjustments or "learns" only if its output array is closest to the target array. Perhaps non-winning workers should be encouraged to increase their distance from the target as well. An objective function that approaches its minimum as the winning worker module's error becomes smaller relative to those of the other workers may be defined as follows.

$$J = \frac{V_w}{\sum_{m \neq w} V_m}$$

where w is the index of the winning worker module, and V_m is the distance of the mth worker's output vector from the target vector. Although many distance measures are possible, a useful one is the sum of the squared differences between output and target after scaling both such that the mid-range of each is zero. That is, for the mth worker module,

$$V_m = \frac{1}{2}\sum_i((y_{mi} - \frac{RO}{2} - MO) - (t_i - \frac{RT}{2} - MT))^2$$

where y_{mi} is the ith output of the module, RO is the range of values in the output array, MO is the minimum of the output array, t_i is the ith target element, RT is the range of values in the target array, and MT is the minimum of the target array.

The function may be rewritten as

$$V_m = \frac{1}{2}\sum_i(y_{mi} - t_i + K)^2$$

where $K = (\frac{RT}{2} + MT) - (\frac{RO}{2} + MO)$.

The gradient with respect to the inputs of the ith output unit of the mth worker module may be defined as

$$\frac{\partial J}{\partial x_{mi}} = \frac{\partial J}{\partial V_m}\frac{\partial V_m}{\partial y_{mi}}\frac{\partial y_{mi}}{\partial x_{mi}}$$

For the winning worker,

$$\frac{\partial J}{\partial x_{wi}} = \frac{1}{\sum_{j \neq w} V_j}(y_{mi} - t_i + K)y_{mi}(1 - y_{mi})$$

For the remaining workers,

$$\frac{\partial J}{\partial x_{mi}} = \frac{-V_w}{(\sum_{j \neq w} V_j)^2}(y_{mi} - t_i + K)y_{mi}(1 - y_{mi})$$

Monitor modules were treated as before; that is, the target was 1.0 for the winning worker's monitor and 0.0 for each of the others, and all learned at every pattern presentation.

The experiment with *w*, *l*, and *ow* classes was repeated using this training algorithm, and the results showed an improvement. Generalization performance was *w* (89.6%), *l* (70.0%), and *ow* (80.0%). Overall performance of 76.6 percent correct was now about one percent better than that of a single multilayer network. Although this result is better than that obtained with the original Minos training procedure, it still is not a significant improvement over the performance of a simpler multilayer network.

6.4 Input Space Transformation Architecture

Preliminary experiments showed that TDNNs are able to map limited regions of the spectrogram space to a phoneme space with variable degrees of accuracy. It is reasonable to assume that intra-class variability present in the original spectrogram data may be somewhat reduced in the new phoneme space resulting from the transformations. A reduction is expected because multilayer networks are able to map disjoint regions of input space to the same target. Such reduced variability may enable a single network to classify the complete phoneme set more easily using input data from the various phoneme subspaces rather than from the original spectrogram space. This section describes and tests the Input Space Transformation (IST) architecture designed to test this conjecture.

An exploratory experiment similar to that described in Section 7 used a single multilayer network to discriminate among 39 phoneme classes obtained by merging some of the classes in Table 1. The merged classes are enclosed in brackets in the following list defining the classes: *p, t, k, b, d, g, (m,em), (n,nx,en), (ng,eng), s, z, ch, th, f, (sh,zh), jh, dh, v, (l,el), r, y, (hh,hv), w, eh, (ao,aa), (uw,ux), (er,axr), ay, ey, aw, (ax,ah,ax-h), (ix,ih), ae, uh, oy, iy, ow, (pau,h#,cl)*. The confusion matrix provided by that network gave an indication of the similarity among the classes. These observed similarities were used to crudely partition the phoneme space into a number of subspaces so that a given TDNN network module could be trained to discriminate only among similar classes. Also on the basis of the confusion matrix, the size of the phoneme space was reduced to 38 by merging classes *d* and *dx*, and *ao* was merged with *aw* rather than *aa*.

Table 5 lists the discriminations learned by each network, and the domain from which the training set was drawn. Figure 5 shows more clearly that the assignment of input regions to modules was organized somewhat hierarchically. Each network was trained with 300 patterns per class. The word "rest" in the table and the figure refers to a class which groups together the remaining classes in the indicated training set domain.

During training, each network was tested periodically to ensure that generalization accuracy was not decreasing due to overlearning. Training stopped when generalization performance failed to improve or a decrease was observed. Then a different, validation test set was used to again measure generalization performance, and these results are reported in Table 6.

The outputs of the 14 TDNN modules identified in Table 5 map the voice spectrogram space into a phoneme space. These outputs (except for some of the "rest" categories) became training patterns for a single multilayer network with 51 inputs, 30 hidden units, and 38 output units. That is, each pattern in the spectrogram space was transformed by all the TDNN modules into a new input pattern in phoneme space. Figure 6 shows the structure of the complete classifier architecture. The multilayer network was trained with 150 patterns per class[6]. After 8500 passes through the training set, a generalization accuracy of 52.4 percent correct classifications over both test sets was achieved. Table 7 lists the 38 classes, with associated frequencies and proportions correct for each test set alone and for the

[6] the maximum number allowed by the available memory

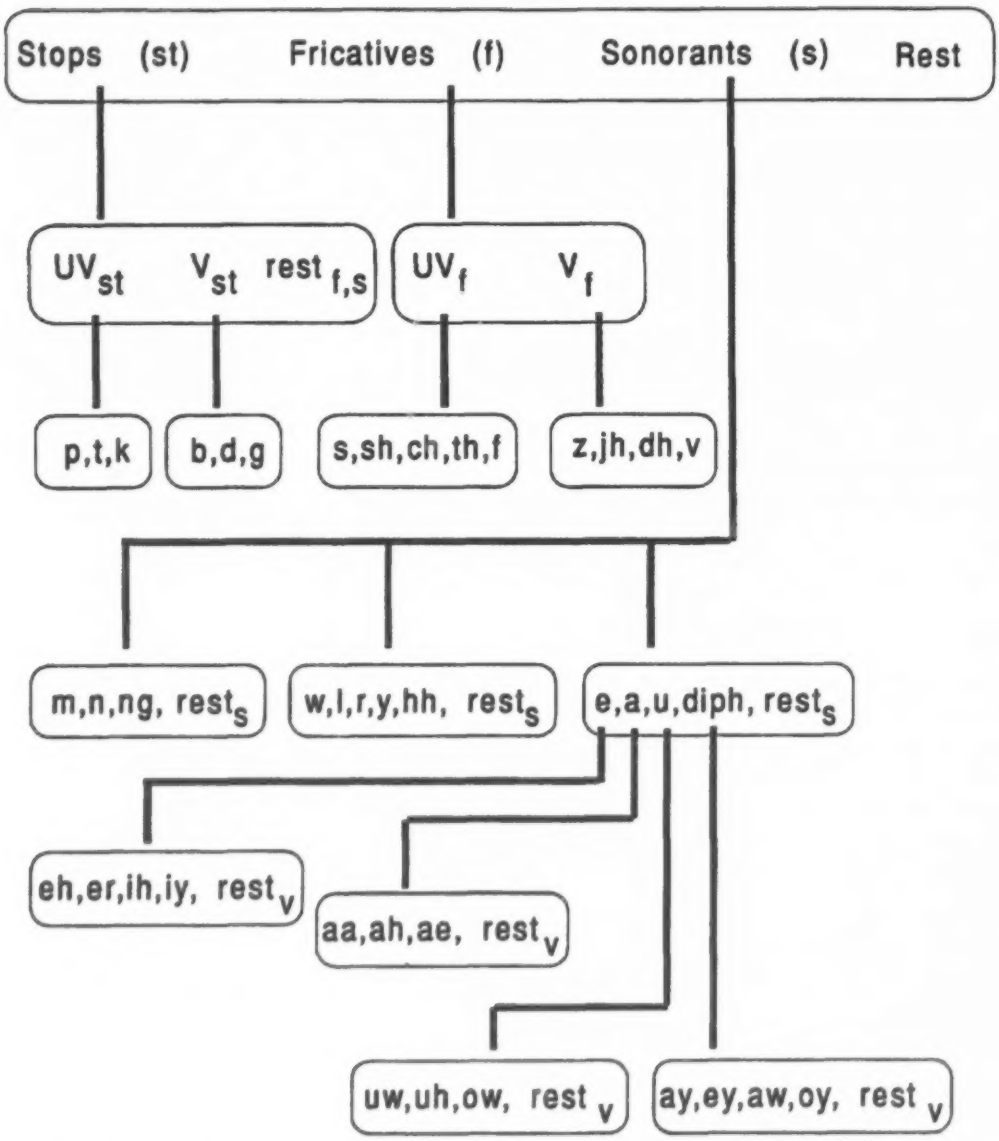


Figure 5: A hierarchical organization of network domains

Table 5: Domains of Input Space Transformation Modules

Output Dimensions	Training Set Domain
Stops, Fricatives, Sonorant, Rest	(unlimited)
Unvoiced, Voiced Stops, Rest	Stops, Fricatives, Sonorants
Unvoiced, Voiced Fricatives, Rest	Fricatives, Stops, Sonorants
P, T, K	Unvoiced Stops
B, D, G	Voiced Stops
S, SH, CH, TH, F	Unvoiced Fricatives
Z, JH, DH, V	Voiced Fricatives
M, N, NG, Rest	Sonorants
W, L, R, Y, HH, Rest	Sonorants
E, A, U, Diphthongs, Rest	Sonorants
EH, ER, IH, IY, Rest	Vowels
AA, AH, AE, Rest	Vowels
UW, UH, OW, Rest	Vowels
AY, EY, AW, OY, Rest	Vowels

Table 6: Performance of Input Space Transformation Modules

Output Dimensions	Percent Correct Generalization
Stops, Fricatives, Sonorant, Rest	78.7, 78.0, 87.6
Unvoiced, Voiced Stops, Rest	88.4, 73.7, 69.0
Unvoiced and Voiced Fricatives	94.9, 75.0
P, T, K	82.2, 86.1, 80.7
B, D, G	79.5, 81.4, 83.3
S, SH, CH, TH, F	82.4, 80.6, 100.0, 78.6, 81.0
Z, JH, DH, V	74.1, 83.3 80.4, 78.6
M, N, NG, Rest	61.4, 74.4, 61.1, 65.1
W, L, R, Y, HH, Rest	74.1, 61.6, 82.7, 87.5, 86.4, 46.0
E, A, U, Diphthongs, Rest	51.1, 43.6, 62.5, 55.2, 68.0
EH, ER, IH, IY, Rest	56.1, 82.5, 55.0, 82.0, 52.8
AA, AH, AE, Rest	83.8, 71.6, 77.5 49.9
UW, UH, OW, Rest	84.4, 85.7, 69.2, 43.8
AY, EY, AW, OY, Rest	69.0, 86.4, 100.0, 50.0, 55.2
Mean Percent Correct	77.7

combined test sets.

The table shows that the *pau* class (representing no speech activity) has both a high frequency of occurrance and a relatively high accuracy score. When this class is ignored in the generalization test to remove its effect from the overall score, a combined accuracy of 45.8 instead of 52.4 percent was obtained.

6.5 Phoneme Filter System Architecture

Another new modular system, which might be considered the IST architecture taken to the limit, consisted of 49 TDNN modules. Each module had one output unit and was trained to discriminate one phoneme class from all the others. After training, each module functioned as a "phoneme filter" since its output signalled the presence of a particular phoneme in the input array. The filter analogy led to the name of Phoneme Filter System (PFS) architecture.

Figure 7 shows the structure of this modular classifier architecture. Each module was trained with 740 exemplars of the phoneme class that it was to identify, as well as 740 exemplars spread evenly across the remaining 48 classes as counter-examples.

In generalization tests, all modules received the same input data, and classification decisions were obtained by identifying the module with the largest output. This architecture resulted in 40.3 percent correct generalization. Table 8 shows the generalization results for Test Sets A and B combined.

7 Experiments with Simple Network Architectures

In order to provide some context for the performance scores obtained from the IST and PFS modular architectures, single multilayer networks and TDNNs were also trained to discriminate among the 38 and 49 phoneme classes.

7.1 The Multilayer Network Performance

A multilayer network with 224 input nodes (14 frames), 20 hidden nodes, and 38 output nodes was trained to discriminate among the 38 phoneme classes. The training set size was 300 patterns per class. Table 9 lists the 38 classes, with associated frequencies and proportions correct for each test set alone and for the combined test sets. It is clear from the table that performance varied considerably among the classes.

A similar experiment was conducted with 49 classes for comparison with the PFS archi-

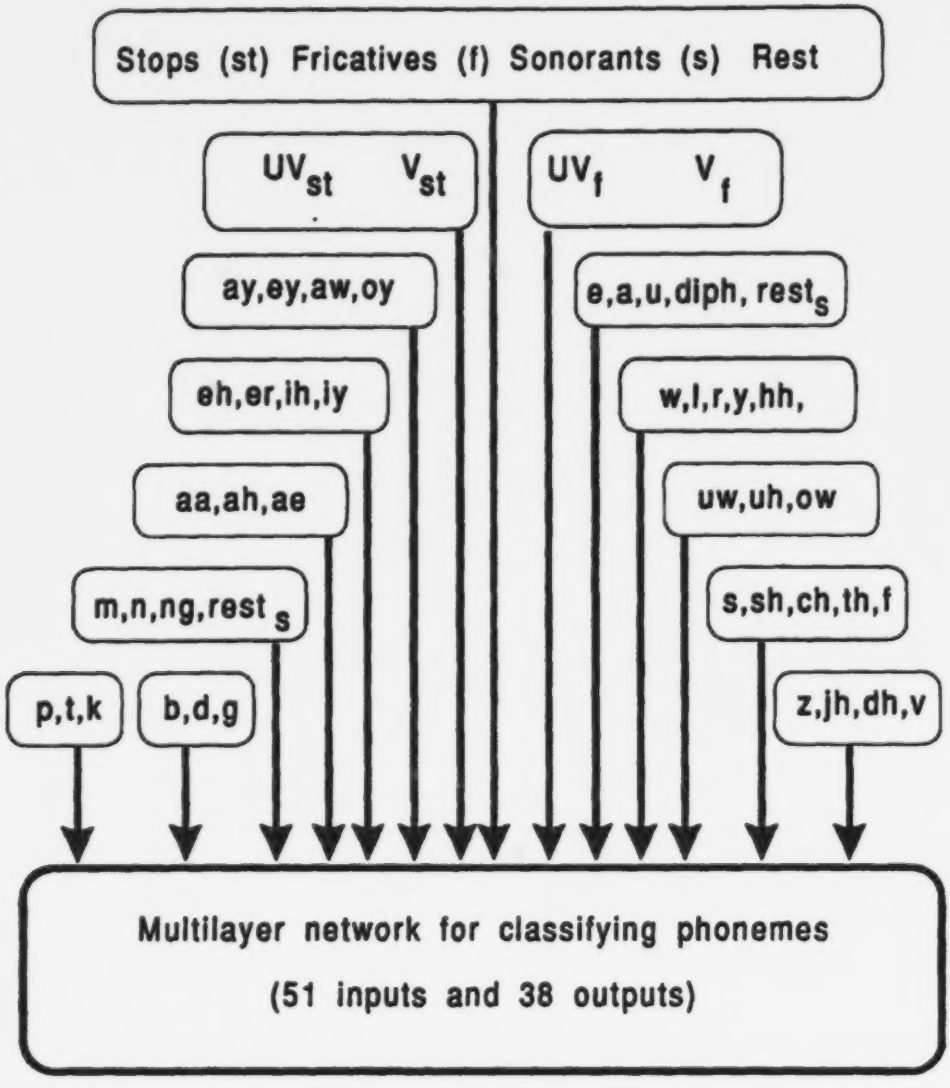


Figure 6: The input space transformation architecture

Table 7: Input Space Transformation Architecture Performance (38 classes)

Class	Set A Freqency	Set A % Correct	Set B Frequency	Set B % Correct	Combined % Correct
p	50	50.0	64	70.3	61.4
t	78	67.9	72	65.3	66.7
k	94	55.3	88	73.9	64.3
b	46	56.5	44	70.5	63.3
d	72	36.1	102	43.1	40.2
g	18	44.4	6	83.3	54.2
s	128	75.8	102	70.6	73.5
sh	25	68.0	36	72.2	70.5
ch	12	75.0	6	100.0	83.3
th	14	35.7	14	50.0	42.9
f	44	81.8	42	76.2	79.1
z	66	60.6	54	46.3	54.2
jh	24	62.5	30	60.0	61.1
dh	52	36.5	46	47.8	41.8
v	46	32.6	42	45.2	38.6
m	86	33.7	83	39.8	36.7
n	152	30.9	172	28.5	29.6
ng	25	60.0	18	66.7	62.8
w	48	47.9	58	56.9	52.8
l	110	50.9	112	43.8	47.3
r	106	44.3	98	34.7	39.7
y	14	50.0	16	68.8	60.0
hh	40	65.0	22	54.5	61.3
eh	66	15.2	66	7.6	11.4
er	72	65.3	80	61.3	63.2
ix	244	24.6	278	20.5	22.4
iy	94	60.6	100	63.0	61.9
aa	34	41.2	48	54.2	48.8
ax	104	24.0	116	14.7	19.1
ae	50	68.0	40	47.5	58.9
uw	32	46.9	32	59.4	53.1
ow	30	26.7	26	34.6	30.4
uh	4	0.0	14	50.0	38.9
ay	48	47.9	42	45.2	46.7
ey	34	61.8	44	65.9	64.1
aw	46	6.5	24	12.5	8.6
oy	10	100.0	6	33.3	75.0
pau	594	78.6	574	76.5	77.6
OVERALL		52.9		51.8	52.4

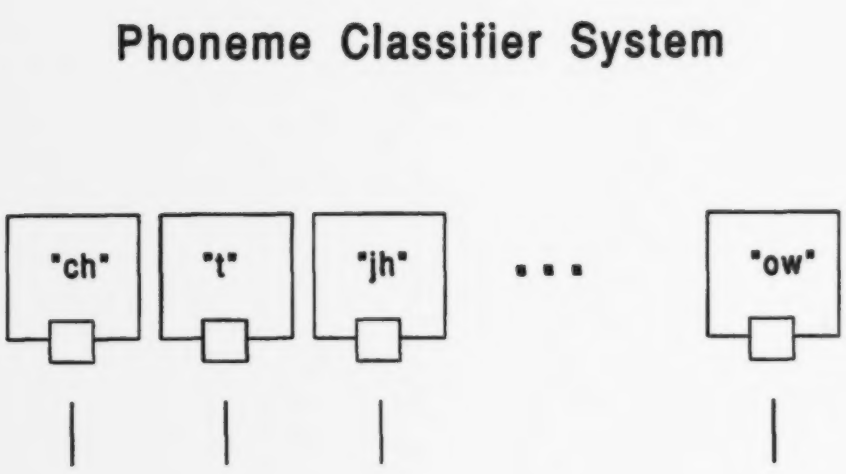


Figure 7: The phoneme filter system architecture

Table 8: Phoneme Filter System Performance (49 classes)

Class	Freqency	% Correct	Class	Frequency	% Correct
p	114	42.1	r	204	26.5
t	150	52.7	y	30	56.7
k	182	46.2	hh	50	52.0
b	90	27.8	hv	12	41.7
d	104	42.3	eh	132	25.0
dx	70	45.7	ao	52	17.3
g	24	54.2	aa	82	39.0
cl	782	52.7	uw	18	16.7
s	230	44.3	er	74	43.2
z	120	49.2	ay	90	12.2
ch	18	16.7	ey	78	20.5
th	28	71.4	aw	18	50.0
f	86	52.3	ax	124	30.6
sh	56	57.1	ix	322	22.0
jh	54	57.4	ih	200	19.0
dh	98	25.5	ae	90	33.3
v	88	56.8	ah	96	6.3
m	164	35.4	uh	18	55.6
n	276	44.6	ux	46	19.6
en	24	50.0	oy	16	43.8
nx	24	50.0	iy	194	24.2
ng	42	33.3	ow	56	12.5
w	106	26.4	axr	78	23.1
l	182	43.4	pau	350	61.7
el	40	25.0			
OVERALL					39.1

Table 9: Multilayer Network Performance (38 classes)

Class	Set A Freqency	Set A % Correct	Set B Frequency	Set B % Correct	Combined % Correct
p	50	56.0	64	56.2	56.1
t	78	67.9	72	58.3	63.3
k	94	53.2	88	54.5	53.8
b	46	76.1	44	75.0	75.6
d	72	22.2	102	20.6	21.3
g	18	77.8	6	50.0	70.8
s	128	74.2	102	66.7	70.9
sh	25	76.0	36	77.8	77.0
ch	12	75.0	6	83.3	77.8
th	14	35.7	14	35.7	35.7
f	44	75.0	42	78.6	76.7
z	66	56.1	54	57.4	56.7
jh	24	70.8	30	63.3	66.7
dh	52	40.4	46	41.3	40.8
v	46	43.5	42	50.0	46.6
m	86	38.4	83	32.5	35.5
n	152	3.3	172	5.8	4.6
ng	25	76.0	18	72.2	74.4
w	48	50.0	58	53.4	51.9
l	110	45.5	112	43.8	44.6
r	106	53.8	98	42.9	48.5
y	14	57.1	16	62.5	60.0
hh	40	62.5	22	63.6	62.9
eh	66	12.1	66	7.6	9.8
er	72	51.4	80	53.8	52.6
ix	244	29.5	278	23.7	26.4
iy	94	57.4	100	63.0	60.3
aa	34	61.8	48	50.0	54.9
ax	104	16.3	116	17.2	16.8
ae	50	58.0	40	57.5	57.8
uw	32	28.1	32	40.6	34.4
ow	30	36.7	26	19.2	28.6
uh	4	25.0	14	50.0	44.4
ay	48	47.9	42	31.0	40.0
ey	34	70.6	44	56.8	62.8
aw	46	8.7	24	8.3	8.6
oy	10	0.0	6	0.0	0.0
pau	594	47.3	574	41.8	44.6
OVERALL		45.0		41.1	43.0

Table 10: Multilayer Network Performance (49 classes)

Class	Freqency	% Correct	Class	Frequency	% Correct
p	114	56.1	r	204	46.1
t	150	54.0	y	30	50.0
k	182	50.5	hh	50	52.0
b	90	70.0	hv	12	8.3
d	104	28.8	eh	132	16.7
dx	70	77.1	ao	52	40.4
g	24	66.7	aa	82	47.6
cl	782	60.2	uw	18	44.4
s	230	59.1	er	74	52.7
z	120	66.7	ay	90	33.3
ch	18	72.2	ey	78	52.6
th	28	39.3	aw	18	55.6
f	86	62.8	ax	124	25.0
sh	56	85.7	ix	322	27.3
jh	54	68.5	ih	200	13.5
dh	98	34.7	ae	90	45.6
v	88	39.8	ah	96	29.2
m	164	33.5	uh	18	38.9
n	276	25.4	ux	46	37.0
en	24	45.8	oy	16	50.0
nx	24	62.5	iy	194	54.1
ng	42	61.9	ow	56	14.3
w	106	46.2	axr	78	21.8
l	182	41.2	pau	350	49.1
el	40	65.0			
OVERALL					45.5

tecture, and the trained network achieved 45.5 percent correct on the generalization test. Table 10 shows the generalization results for Test Sets A and B combined.

7.2 The Time Delay Neural Network Performance

A single TDNN network also learned to discriminate among the 38 classes using 300 patterns per class as training data. The test performance results are presented in Table 11.

Similarly, a TDNN was trained to discriminate among the 49 classes, and achieved 45.7 percent correct on the generalization test. Table 12 shows the generalization results for Test Sets A and B combined.

Table 11: TDNN Performance (38 classes)

Class	Set A Freqency	Set A % Correct	Set B Frequency	Set B % Correct	Combined % Correct
p	50	56.0	64	62.5	59.6
t	78	59.0	72	62.5	60.7
k	94	51.1	88	44.3	47.8
b	46	69.6	44	79.5	74.4
d	72	30.6	102	19.6	24.1
g	18	22.2	6	16.7	20.8
s	128	71.9	102	65.7	69.1
sh	25	64.0	36	77.8	72.1
ch	12	83.3	6	100.0	88.9
th	14	35.7	14	42.9	39.3
f	44	70.5	42	66.7	68.6
z	66	66.7	54	66.7	66.7
jh	24	58.3	30	63.3	61.1
dh	52	38.5	46	30.4	34.7
v	46	58.7	42	40.5	50.0
m	86	34.9	83	48.2	41.1
n	152	21.1	172	34.3	28.1
ng	25	64.0	18	50.0	58.1
w	48	33.3	58	46.6	40.6
l	110	42.7	112	42.0	42.3
r	106	33.0	98	38.8	35.8
y	14	64.3	16	68.8	66.7
hh	40	15.0	22	31.8	21.0
eh	66	21.2	66	12.1	16.7
er	72	52.8	80	60.0	56.6
ix	244	31.6	278	30.2	30.8
iy	94	53.2	100	56.0	54.6
aa	34	52.9	48	54.2	53.7
ax	104	17.3	116	10.3	13.6
ae	50	64.0	40	40.0	53.3
uw	32	25.0	32	43.8	34.4
ow	30	33.3	26	30.8	32.1
uh	4	0.0	14	21.4	16.7
ay	48	52.1	42	52.4	52.2
ey	34	61.8	44	59.1	60.3
aw	46	10.9	24	8.3	10.0
oy	10	30.0	6	33.3	31.2
pau	594	58.8	574	57.7	58.2
OVERALL		46.2		46.0	46.1

Table 12: TDNN Performance (49 classes)

Class	Freqency	% Correct	Class	Frequency	% Correct
p	114	67.5	r	204	38.2
t	150	56.7	y	30	53.3
k	182	58.8	hh	50	40.0
b	90	66.7	hv	12	8.3
d	104	36.5	eh	132	3.8
dx	70	70.0	ao	52	44.2
g	24	54.2	aa	82	34.1
cl	782	51.4	uw	18	38.9
s	230	64.8	er	74	58.1
z	120	70.8	ay	90	36.7
ch	18	72.2	ey	78	59.0
th	28	39.3	aw	18	27.8
f	86	69.8	ax	124	25.0
sh	56	73.2	ix	322	18.6
jh	54	64.8	ih	200	21.5
dh	98	12.2	ae	90	44.4
v	88	39.8	ah	96	16.7
m	164	28.7	uh	18	16.7
n	276	34.8	ux	46	43.5
en	24	41.7	oy	16	25.0
nx	24	45.8	iy	194	39.7
ng	42	64.3	ow	56	25.0
w	106	50.9	axr	78	26.9
l	182	42.9	pau	350	84.6
el	40	67.5			
OVERALL					45.7

Table 13: Summary of Generalization Results for W, L, and OW

Network	W	L	OW
MLP	91.7	67.3	80.0
SYSEX	90.9	56.0	75.0
MINOS	81.3	61.8	76.7
MINOS +	89.6	70.0	80.0

8 Conclusions

The problems with scaling up the size of back-propagation networks has resulted in a number of proposals for the design of modular network systems. However, many such interesting theoretical ideas are often tested only with well understood and controlled problems. A solution that appears promising under those conditions may not, however, be very useful under much more variable conditions. The objective of this program of research was to evaluate some of these ideas in the context of the real-world problem of speaker-independent speech recognition where intra-class variability of speech data is known to be quite large.

The Meta-Pi, system of experts, and Minos architectures all basically attempt to do the same thing; that is, to form criteria useful for identifying regions of the input space. Under the conditions tested here, however, these modular systems could not perform better than a single network trained to handle the whole input space (see Table 13). This may suggest that the overlap in class distributions was such that region boundaries could not be clearly identified by the Meta-Pi module, the system of experts gating module, or Minos monitor modules. It is also possible that the simpler networks may have performed optimally with the given data, so no advantage could be gained by using the more complex architectures.

Two new IST and PFS architectures were designed to circumvent practical constraints on memory and processing speed during training, as well as any detrimental effects of crosstalk arising from the large number of classes. Again, the obtained generalization performance for each system was roughly equivalent to that of a single TDNN or a multilayer network. Table 14 summarizes these results. The IST system's performance was shown to be slightly inflated by a biased response to the high frequency *silence* class. The performance of the PFS architecture might be improved somewhat by weighting the outputs appropriately.

Since the generalization behaviours of the TDNN and multilayer network were not significantly different from those of the modular architectures, crosstalk resulting from the large number of classes did not appear to be a problem. These experiments tend to confirm that, for this data set, the size of the input and output spaces are less of a problem for the learning algorithm than variability of the training data.

Because the TDNN and multilayer network appear equally capable of identifying speech tokens that are appropriately aligned in the input window, the network of choice will depend on other factors. The current preference is for the TDNN network because it is less sensitive

Table 14: Summary of Generalization Results by Network Types

# Classes	IST	PFS	MLP	TDNN
38	52.4		43.0	46.1
49		39.1	45.5	45.7

to input token alignment, and produces less variable output when continuous speech data is passed through its input window.

References

[1] *American standard specification for octave, half-octave, and third-octave band filter sets.* American National Standards Institute, New York, NY, s1.11-1971 edition, 1971.

[2] J.S. Bridle, M.D. Brown, and R.M. Chamberlain. An algorithm for connected word recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 899–902, IEEE, New York, Paris 1982, 1988.

[3] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1:372–381, 1989.

[4] S.J. Cox. Hidden Markov models for automatic speech recognition: theory and application. *British Telecom Technical Journal*, 6:105–115, 1988.

[5] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[6] S.E. Fahlman. Fast-learning variations on back-propagation: an empirical study. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51, Morgan Kaufmann, San Mateo, Pittsburg 1988, 1988.

[7] J.S. Garofolo. *The structure and format of the DARPA TIMIT CD-ROM prototype.* National Institute of Standards and Technology, Gaithersburg, MD, 1988.

[8] J.B. Hampshire and B. Pearlmuter. Equivalence proofs for multi-layer perceptron classifiers and the Bayesian discriminant function. In D.S. Touretzky, J.L. Elman, T.J. Sejnowski, and G. Hinton, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 159–172, Morgan Kaufmann, San Mateo, La Jolla, 1990, 1988.

[9] J.B. Hampshire and A.H. Waibel. *The Meta-Pi network: Building distributed knowledge representations for robust pattern recognition.* Technical Report CMU-CS-89-166, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1989.

[10] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[11] R.A. Jacobs, M.I. Jordan, and A.G. Barto. Task decomposition through competition in a modular connectionist architecture: the what and where vision tasks. *Cognitive Science*, 15:219–249, 1991.

[12] K.J. Lang and G.E. Hinton. *The development of the time-delay neural network architecture for speech recognition.* Technical Report CMU-CS-88-152, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1988.

[13] K.J. Lang, A.H. Waibel, and G.E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[14] K-F Lee. On large-vocabulary speaker-independent continuous speech recognition. *Speech Communication*, 7:375–379, 1988.

[15] K-F Lee, H-S Hon, and R. Reddy. An overview of the SPHINX speech recognition system. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38:35–45, 1990.

[16] H.C. Leung and V.W. Zue. Some phonetic recognition experiments using artificial neural networks. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 422–425, IEEE, New York, New York 1988, 1988.

[17] S.J. Nowlan. *Competing experts: An experimental investigation of associative mixture models.* Technical Report CRG-TR-90-5, Department of Computer Science, University of Toronto, Toronto, Canada, 1990.

[18] S.J. Nowlan and G.E. Hinton. Evaluation of adaptive mixtures of competing experts. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 550–557, Morgan Kaufmann, San Mateo, Denver 1991, 1991.

[19] D.C. Plaut and G.E. Hinton. Learning sets of filters using back-propagation. *Computer Speech and Language*, 2:35–61, 1987.

[20] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, January:4–16, 1986.

[21] A.J. Robinson and F. Fallside. *Phoneme recognition from the TIMIT database using recurrent error propagation networks.* Technical Report CUED/F-INFENG/TR.42, University of Cambridge, Cambridge, England, 1990.

[22] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, chapter 8, pages 318–362, MIT Press, Cambridge, 1986.

[23] H. Sawai, A. Waibel, M. Miyatake, and K. Shikano. Spotting Japanese CV-syllables and phonemes using time-delay neural networks. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 25–28, IEEE, New York, Glasgow 1989, 1989.

[24] F.J. Smieja. Multiple network systems (Minos) modules: Task division and module discrimination. In *Proceedings of the 8th AISB90 Conference on Artificial Intelligence*, Leeds 1991, 1991.

[25] W.C. Treurniet, M.J. Hunt, C. Lefevbre, and Z. Jacobson. Phoneme recognition with a neural network: comparisons of acoustic representations including those produced by an auditory model. In *IEEE International Conference on Neural Networks*, page 320, IEEE, New York, San Diego 1988, 1988.

[26] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:328–339, 1989.

[27] A. Waibel, H. Sawai, and K. Shikano. Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:1888–1898, 1989.

[28] J.G. Wilpon, L.R. Rabiner, and A. Bergh. Speaker-independent isolated words recognition using a 129-word airline vocabulary. *Journal of the Acoustical Society of America*, 72:390–396, 1982.

[29] S.R. Young, A.G. Hauptmann, W.H. Ward, E.T. Smith, and P. Werner. High level knowledge sources in usable speech recognition systems. *Communications of the ACM*, 32:183–194, 1989.

This page intentionally left blank.

NEAREST-NEIGHBOUR SEARCH IN CONSTANT TIME: AN ALTERNATIVE TO NEURAL NETWORKS FOR PATTERN MATCHING APPLICATIONS

Pierre Zakarauskas and John M. Ozard

Defence Research Establishment Pacific, FMO Victoria, B.C. Canada, V0S 1B0.

Contact Pierre Zakarauskas, (604) 363-2879

ABSTRACT

In this paper we present an alternate algorithm to the use of neural networks for pattern matching in continuous space, and present performance estimates as a function of the dimensionality of the search space. Neural networks divide decision space with hyperplanes whose positions are adjusted during training, usually through gradient descent. The decision regions thus obtained are not necessarily optimal for noisy data. Another drawback of neural networks is the rapid increase in training time with the size of the problem, leading to the quasi impossibility of training them for very large problems. In order to obtain robust pattern matching for problems where the number of patterns becomes very large, we propose the use of a fast nearest-neighbour algorithm. This algorithm finds the training pattern closest to a test pattern, in a time which is asymptotically constant with the total number of patterns. The algorithm first partitions the search space, and finds which partition each of the training pattern falls into. When a test pattern is supplied, one compares it to all patterns within the partitions which are within a given distance R to the test pattern. If the distance d between the closest pattern found during the first pass and the test pattern is less than R, then the search stops. If not, then a second pass is done using d as a search parameter to select partitions. This algorithm guarantees that the nearest-neighbour is found. The mean number of operations done to find the nearest-neighbour is presented as a function of the density of patterns per partition and the dimensionality of the search space. A global minimum is shown to exist for the mean number of operations at slightly less than one pattern per partition.

CONTENT

Pattern matching and classification

Nearest-neighbour rule

Fast nearest-neighbour search
-Necessity of searching more than one bucket

Algorithms analyzed:

- Product partitioning
- k-d tree partitioning
- Ordered partitioning
  - Cylindric
  - Limited

Analysis of performance

Pattern matching and classification

- Neural networks often used for performing pattern matching tasks

- Given a distribution of labelled patterns x_q find class C_i that x_q is most likely to belong to.

- Multilayered feedforward neural networks divide parameter space with decision hyperplanes.
  - Efficient, but difficult to find correct planes
  - Scales extremely badly with size of problem
  - Classification not optimum in high noise situations

- Classic pattern matching techniques more robust, and maybe more economical than neural networks in certain cases.

Nearest-neighbour rule

Classification on the basis of the class most heavily represented among the k-nearest neighbors (kNN) to the test pattern is asymptotically Bayes' risk efficient.

$$\rho \leq r_{NN} \leq 2\rho$$

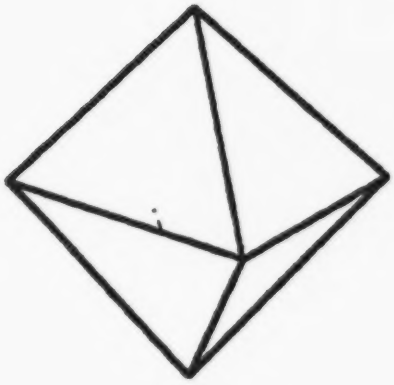
The probability of the NN of being in the same bucket as the test pattern is

$$p_{NN} = \left(1 + n^{-1/d}\right)^{-d}.$$

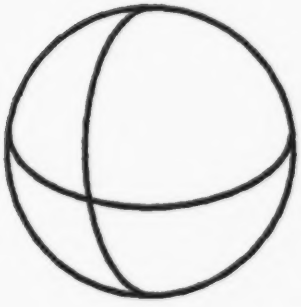
The Minkowski p-metric is measure of distance D_p between two points , and is defined as

$$D_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p}$$

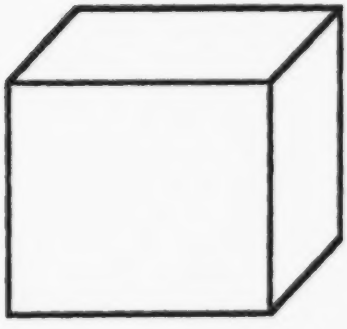
Shape of hypersurface for different Minkowski p-metrics



$p = 1$



$p = 2$



$p = \infty$

The hypervolume of the search p-sphere is

$$V_r = K_{p,d} r^d$$

where $K_{p,d}$ is a constant determined by p and d.

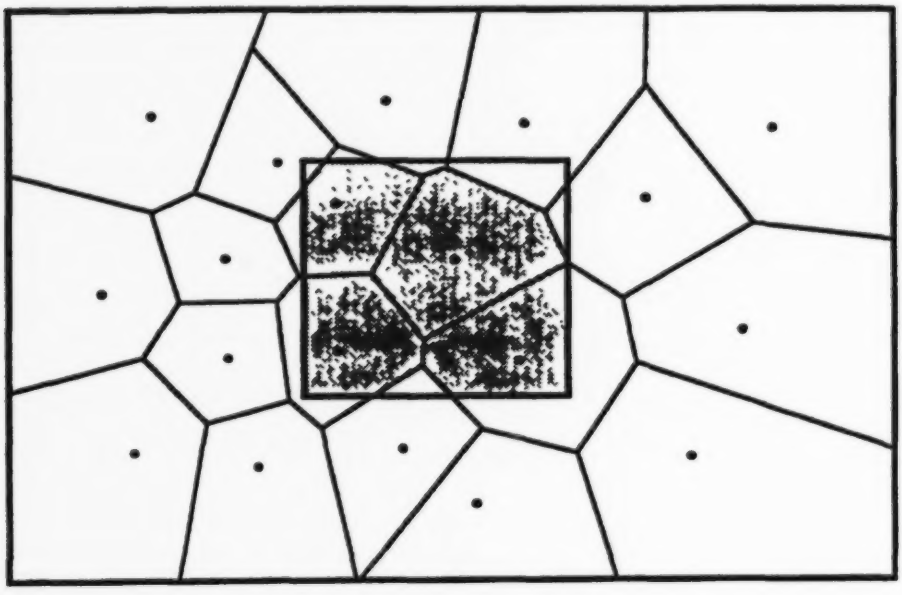
$$K_{1,d} = \frac{2^d}{d!}$$

$$K_{2,d} = \frac{\pi^{d/2}}{\Gamma\left(d/2+1\right)},$$

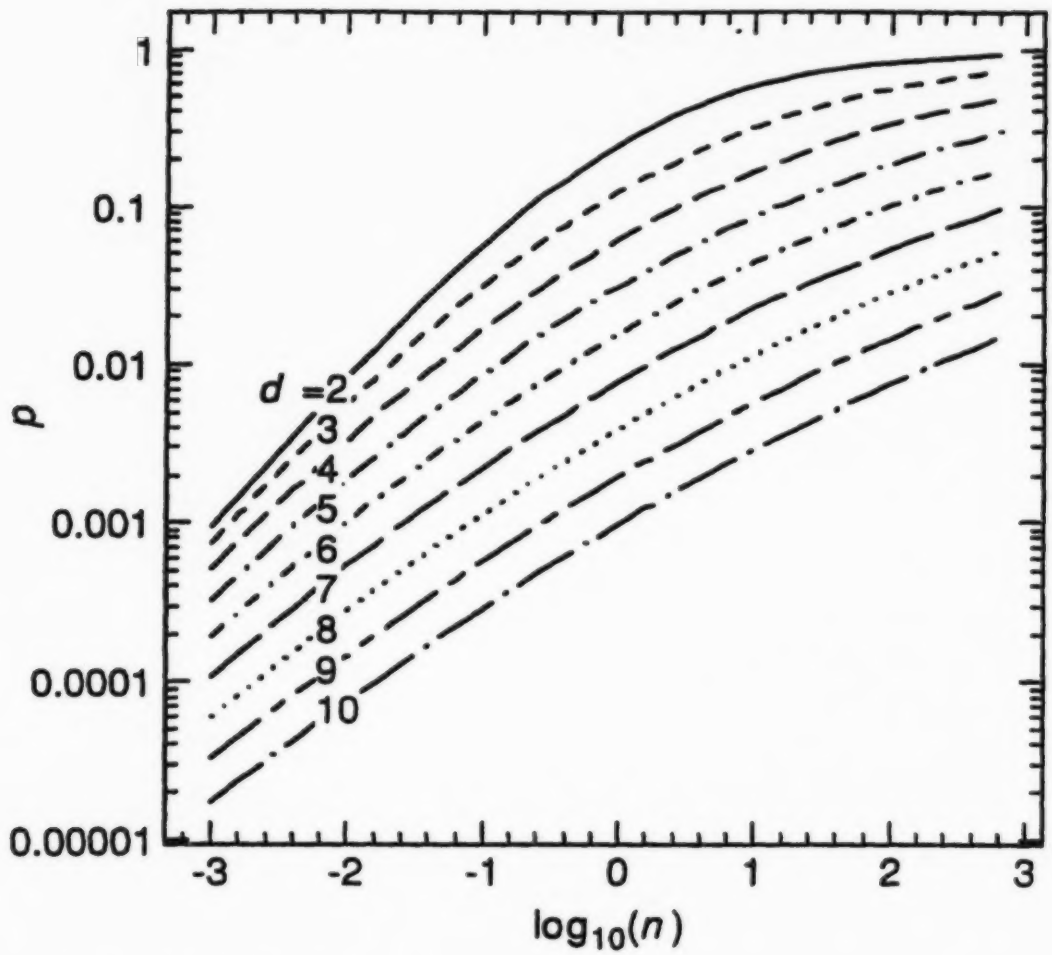
and

$$K_{\infty,d} = 2^d$$

Points having a nearest-neighbour within a given cube



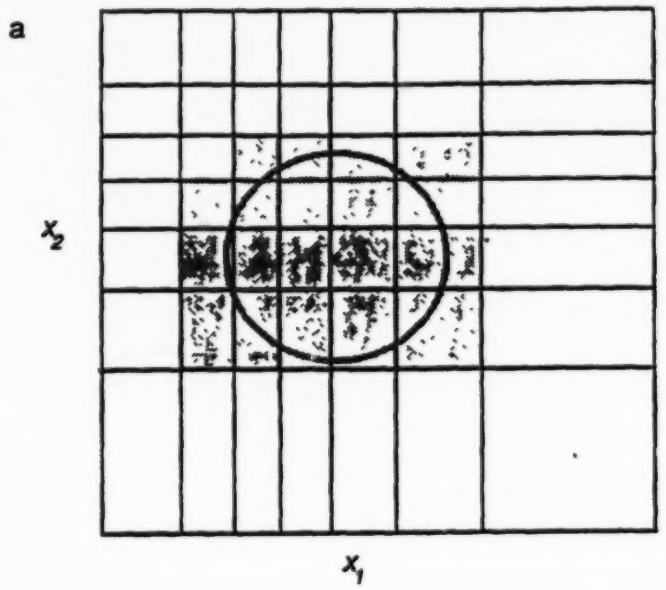
Probability of the NN being in the same bucket as the test pattern

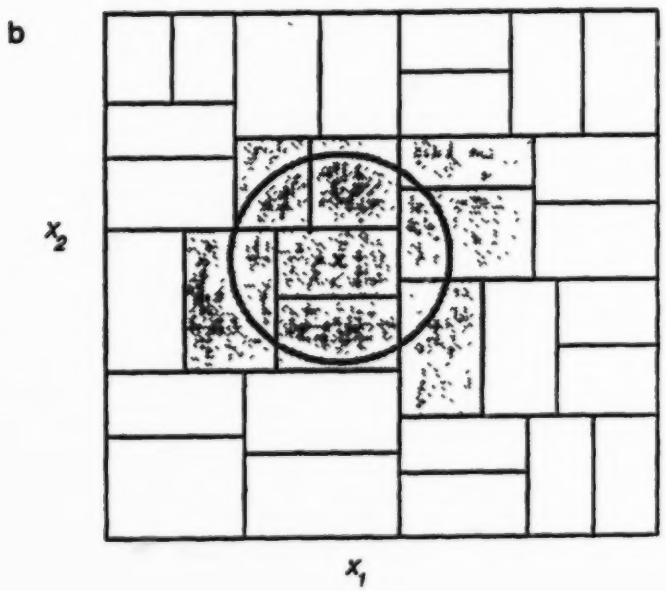


n = # patterns/bucket

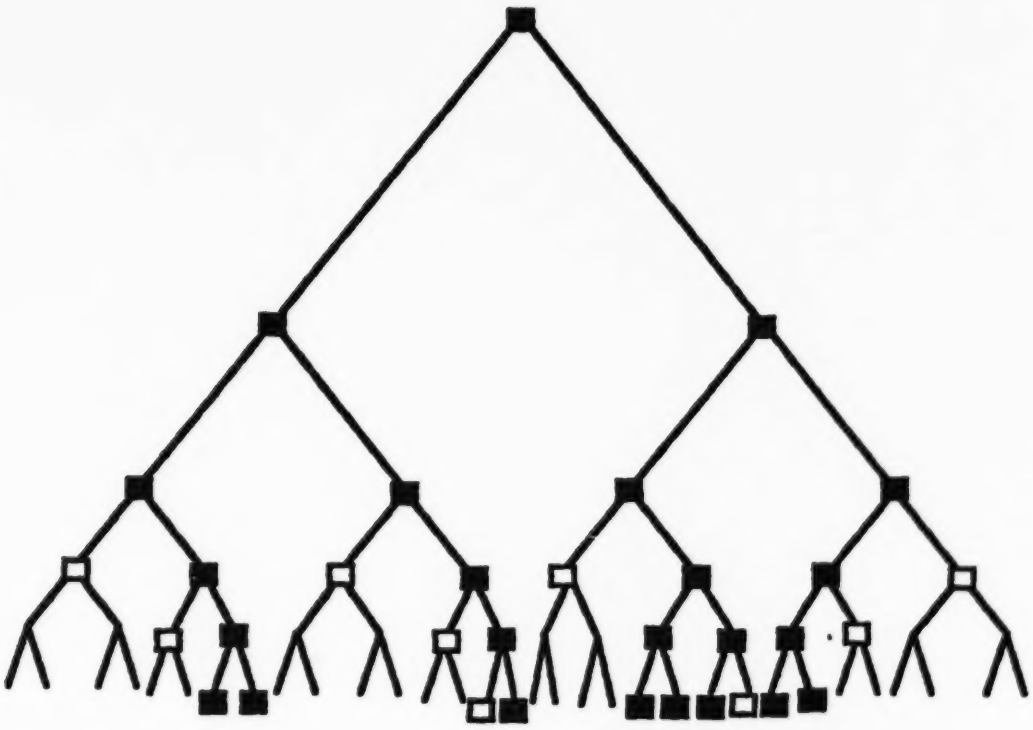
d = # search space dimensions

The ensemble of buckets the touch the search sphere for: (a) product partitioning, (b) k-d tree

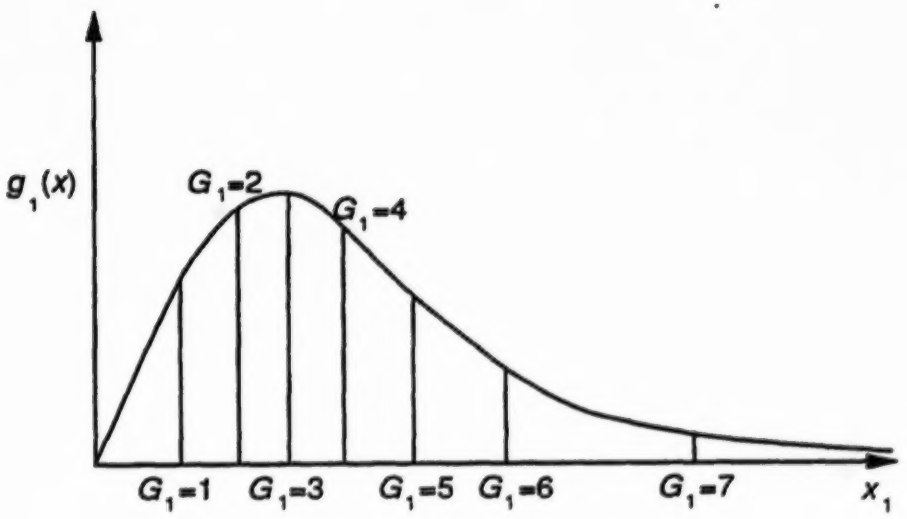




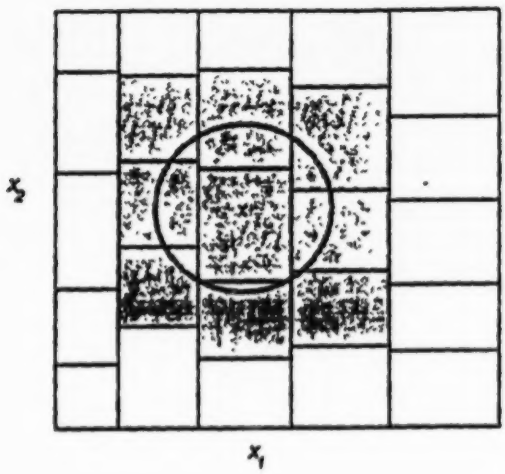
Binary search tree for k-d tree



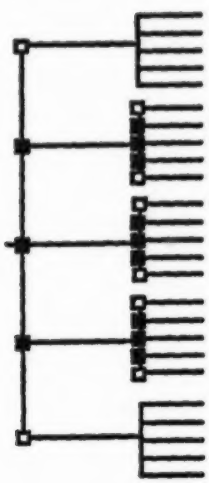
Pattern density function for placement of partitions in product partitioning



Ordered partitioning



Search tree for ordered partitioning



Ω_d = probability for a p-sphere of radius r of hitting a given bucket in a2-dimensional space

$$\Omega_2 = w_1 w_2 + 2r(w_1 + w_2) + \pi r^2$$

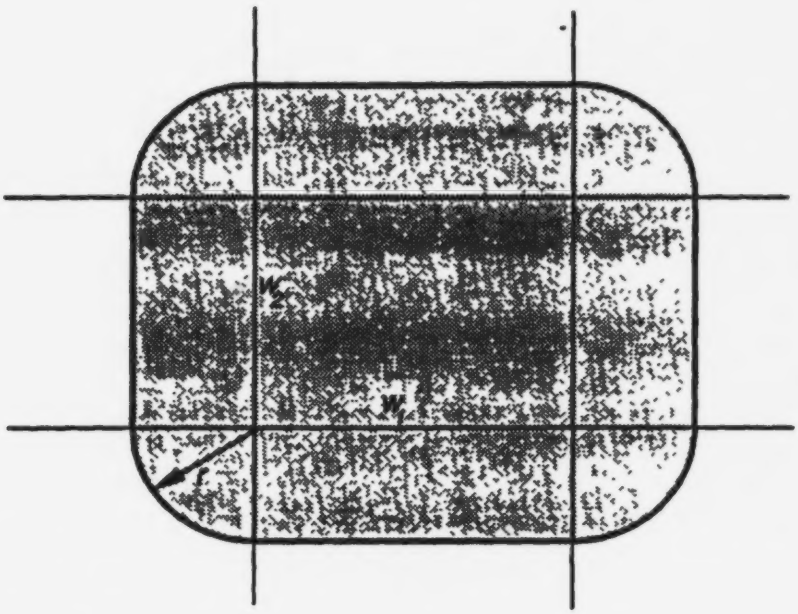
total number of buckets partitioning the p-sphere is

$$B_2 = \frac{\Omega_2}{w_1 w_2} = 1 + 2r\frac{(w_1 + w_2)}{w_1 w_2} + \frac{\pi r^2}{w_1 w_2}.$$

In d-dimensional space

$$B_d = \sum_{i=0}^{d} \frac{K_{p,i} r^i}{w^i} \binom{d}{i}$$

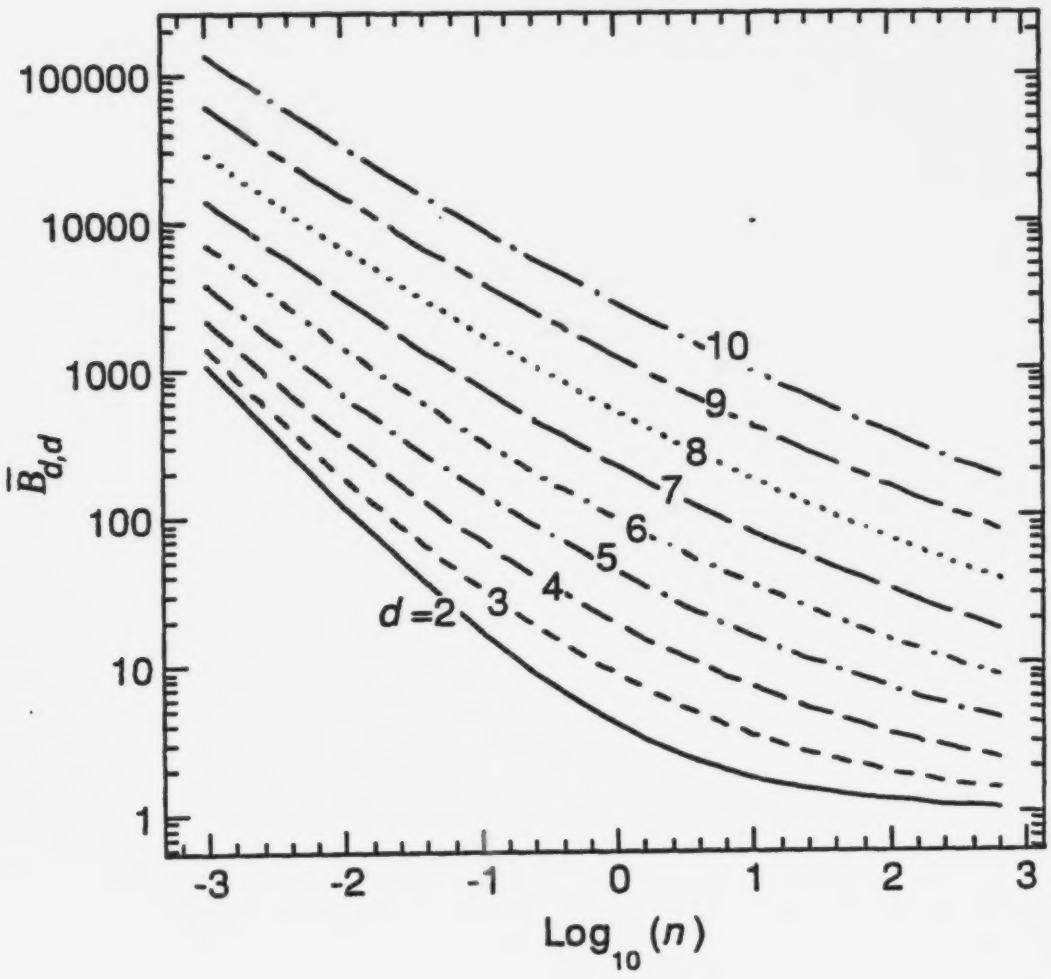
Region within a distance r of a bucket

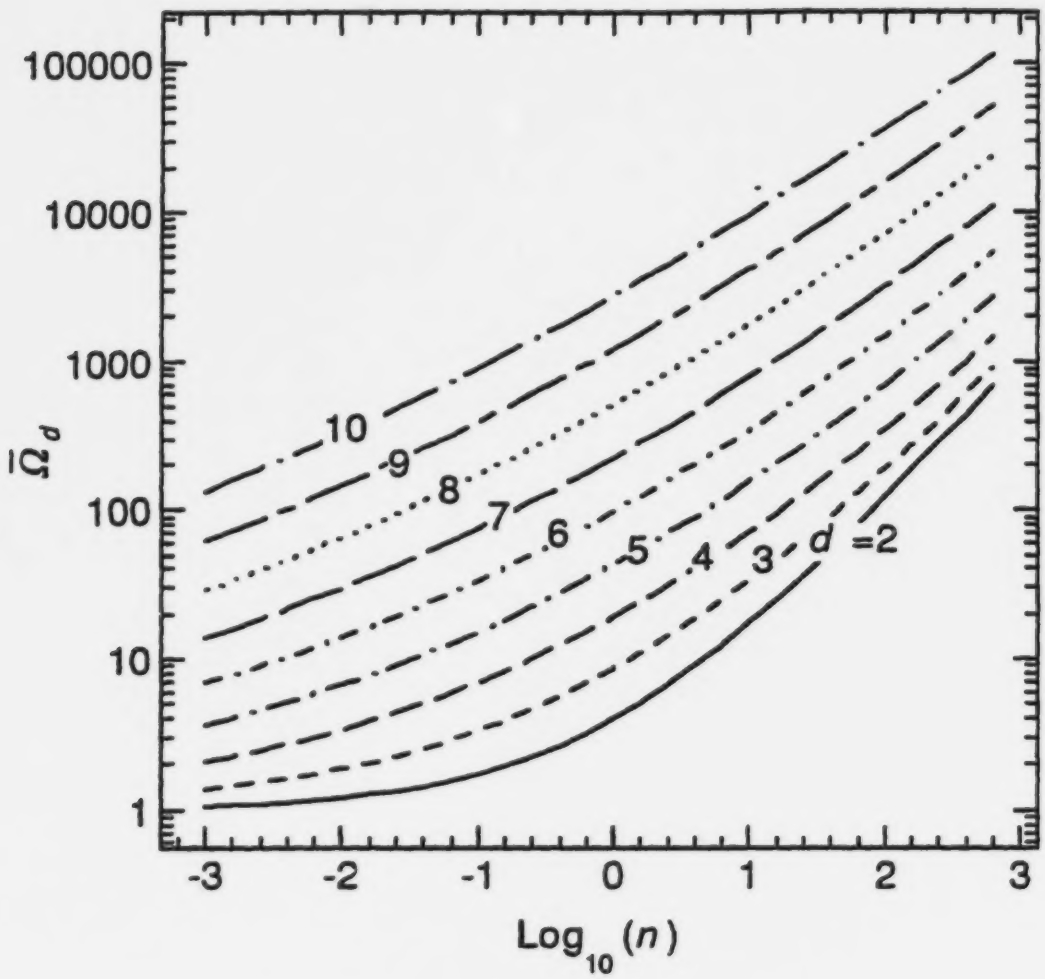


Expected number of j-dimensional buckets to search and patterns to compare

$$\overline{B}_{j,d} = E\left[B_j\right] = \sum_{i=0}^{j} \frac{\left(nK_{p,d}\right)^{-i/d} K_{p,i}}{(k-1)!} \binom{j}{i} \Gamma\left(\frac{i+dk}{d}\right)$$

$$\overline{\Omega}_d = n\overline{B}_{j,d}$$




100000
10000
1000
100
10
1
Ω_d
10
9
8
7
6
5
4
3
d =2
-3
-2
-1
0
1
2
3
Log$_{10}$ (n)

Total cost C_t:

$$C_t = C_o + \overline{\Omega}_d C_d$$

Product partitioning

$$C_o = \overline{B}_{d\text{-}1,d} C_r + \overline{B}_d C_b$$

Cylindric ordered partitioning

$$C_o = \sum_{i=1}^{d-1} \left(\overline{B}_{i,d} + 2\overline{B}_{i-1,d}\right) C_{d,i} + C_b \overline{B}_{d-1,d}.$$

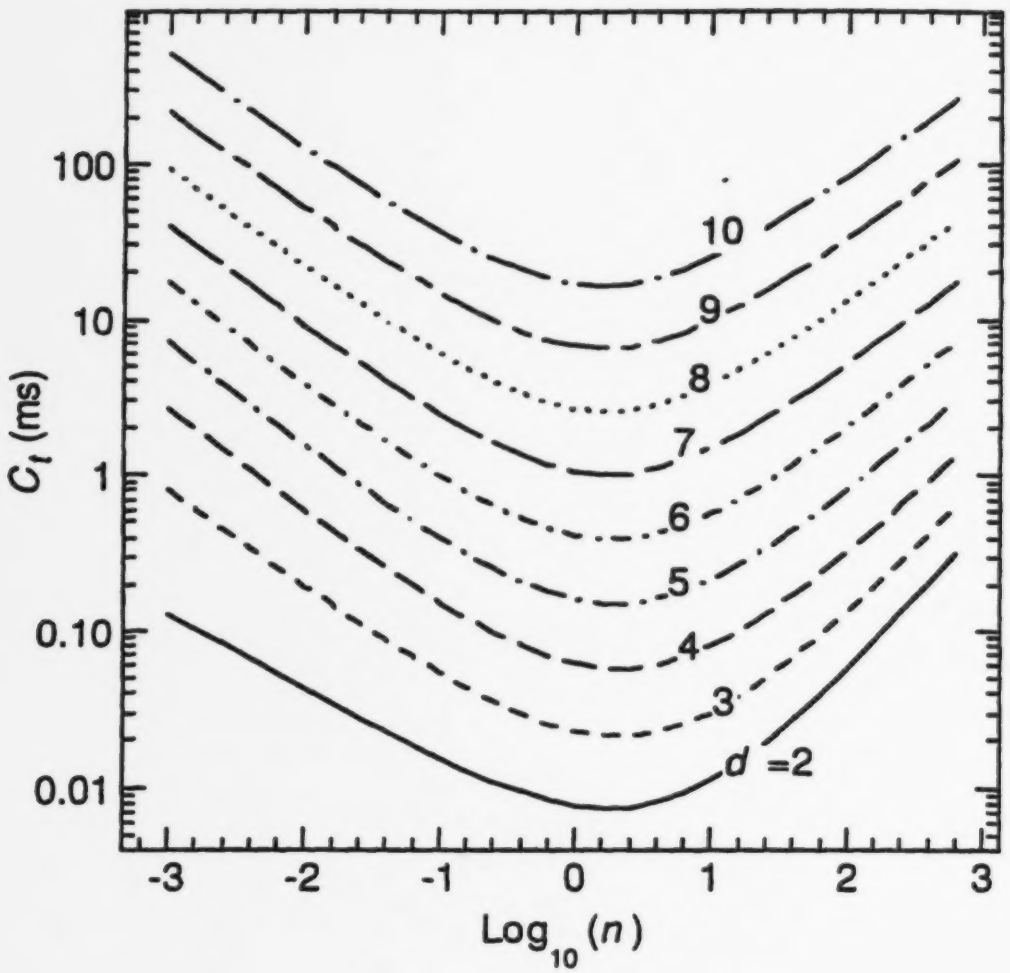
Limited ordered partitioning

$$C_o = \sum_{i=1}^{d-1} \left(\overline{B}_{i,d} + 2\overline{B}_{i-1,d}\right) C_{d,i} + C_b \overline{B}_{d-1,d} + C_r \overline{B}_{d-1,d}$$

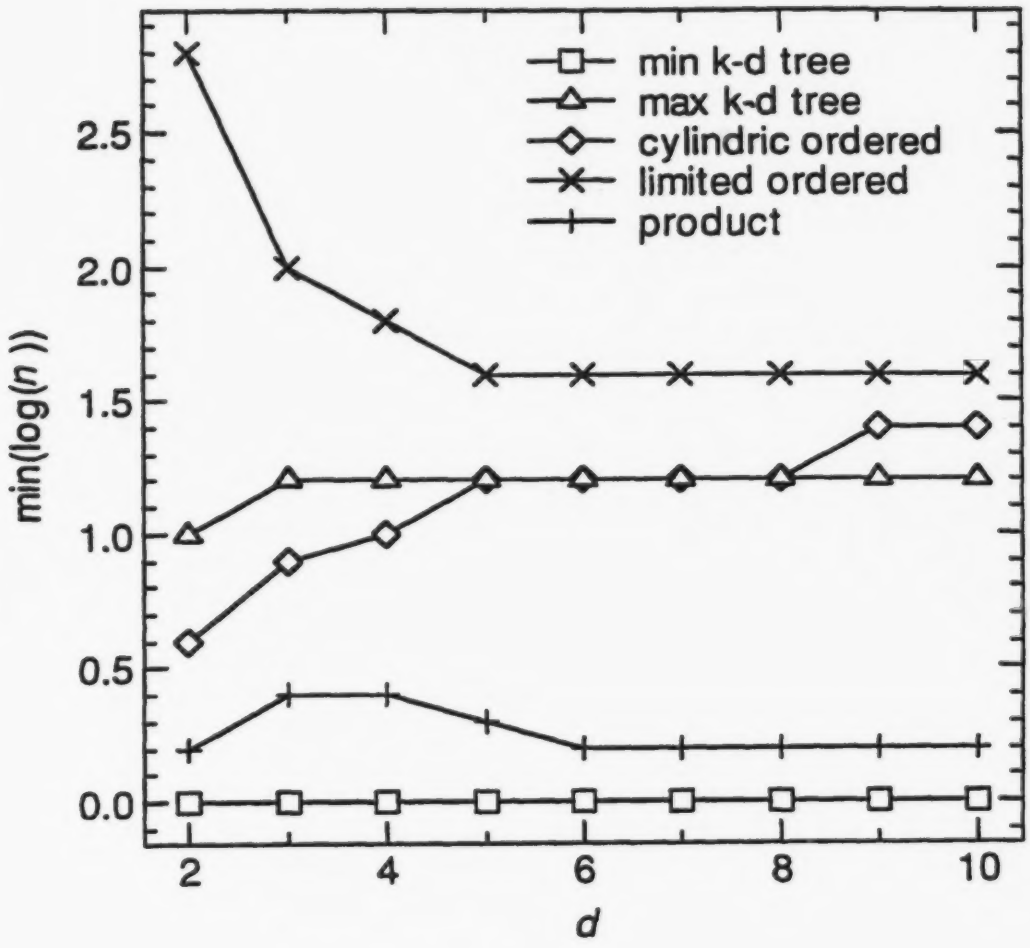
k-d tree

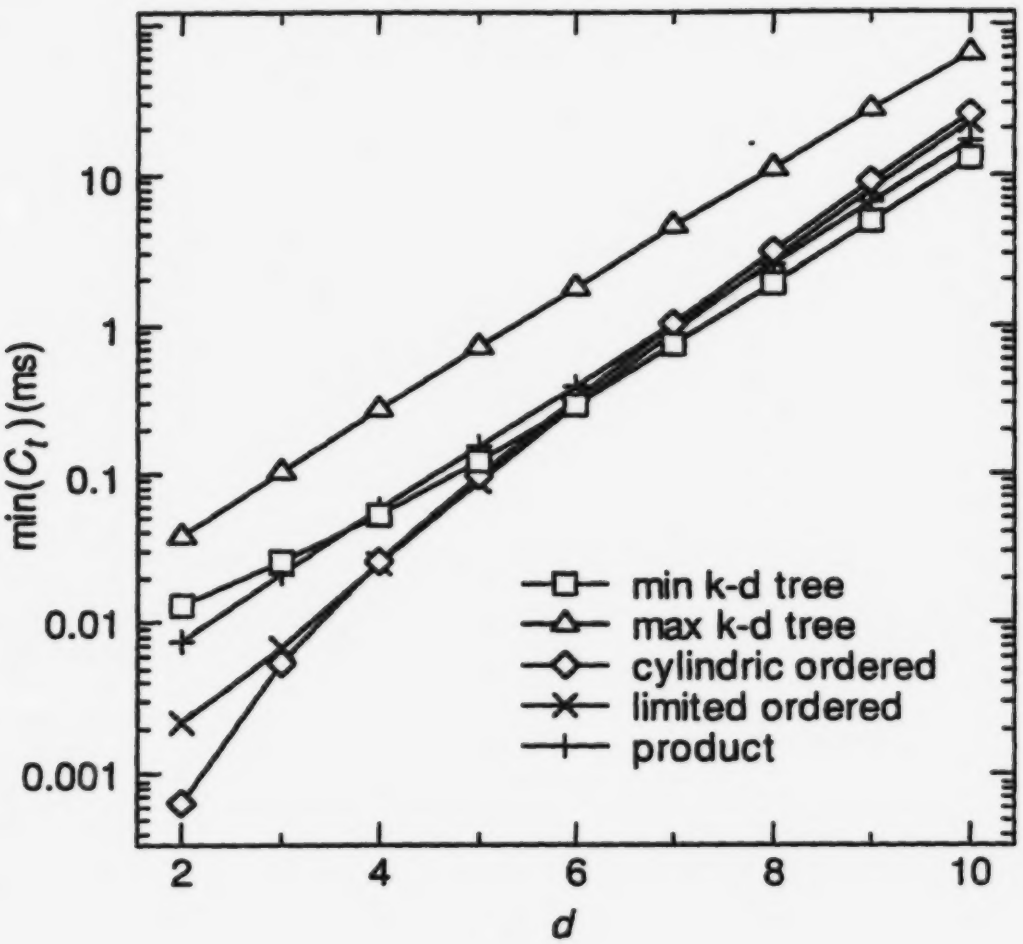
$$\min(N_i) = \min\left(\frac{\overline{B}_{d,d}}{2^{M-i+1}} + 1,\ 2^{i-1}\right)$$

$$\max(N_i) = \min\left(\overline{B}_{d,d},\ 2^{i-1}\right) \qquad M = \text{int}(\log_2 N) + 1$$



product partitioning


min k-d tree
max k-d tree
cylindric ordered
limited ordered
product
min(log(n))
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
d


10
1
0.1
0.01
0.001
min(C_t) (ms)
2
4
6
8
10
d
min k-d tree
max k-d tree
cylindric ordered
limited ordered
product

CONCLUSIONS

- Nearest-neighbour rule states that kNN is near the optimum classification rule, especially in high noise.

- Fast nearest-neighbour search divides search space into buckets, and pre-determine which pattern falls into which bucket. Equivalent to one-pass training, no generalization.

- Only those buckets close to the test pattern are searched.

- The optimum bucket size was calculated.

- Search time is asymptotically constant with number of patterns.

- In high d-space, all algorithms compared perform more or less the same. Not true for $d \leq 4$.